МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
Вычислительной техники

Кафедра
«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Вычислительные машины, комплексы, системы и сети»

БАКАЛАВРСКАЯ РАБОТА

на тему

РАЗВИВАЮЩЕЕ ПРИЛОЖЕНИЕ НА UNITY В ЖАНРЕ «ГОЛОВОЛОМКА»

Студент _____ Богатков К.А.
(подпись, дата) 19.06.23 (ФИО полностью)

Руководитель _____ 19.06.23 И. В. Акифьев
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ 19.06.23 А. С. Бычков
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 02.06.2023 г. № 14 _____)

Заведующий кафедрой _____ М.А. Митрохин
(подпись) (фамилия, инициалы)

Работа защищена с отметкой удовлетворительно (протокол заседания ГЭК от 20.06.2023 г. № 7)

Секретарь ГЭК _____ Е.И. Гурин
(подпись) (фамилия, инициалы)

Пенза 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
_____ М.А. Митрохин
«26» 04 2023 г.

ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ

1. Студент Кочетков Кирилл Алексеевич гр.19ВВВ1 факультета вычислительной техники.

2.Тема работы Развивающее приложение на Unity в жанре «Головоломка».

Тема утверждена приказом ПГУ № 458 от "26" апреля 2023г.

3. Руководитель бакалаврской работы к.т.н., доцент, каф. ВТ ПГУ Акифьев И.В.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

В ходе выполнения выпускной квалификационной работы необходимо разработать приложение на Unity в жанре «Головоломка», отвечающий всем требованиям функциональности, дизайна и структуры современного игрового приложения. Главным требованием является понятный интерфейс. Красивый внешний вид. Написание оптимизированных скриптов для игры.

5. Перечень подлежащих разработке вопросов. Требуется произвести анализ предметных областей: изучить существующие игры головоломки, выделить их достоинства и недостатки. Выбрать средства для разработки, которые подойдут для создания игры. Написать сюжет для игры в жанре «Головоломка». Выбрать вспомогательные средства разработки.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	32	26.04.2023 – 28.05.2023	
2	Анализ требований	32	29.04.2023 – 30.05.2023	
3	Создание спрайтов и графики для игры	30	1.04.2023 – 3.05.2022	
4	Выбор технологий, применяемых при разработке	40	4.05.2023 – 8.05.2023	
5	Реализация скриптов для объектов и персонажей	66	9.05.2023 – 15.05.2023	
6	Разработка основной логики игры	70	16.05.2023 – 23.05.2023	
7	Оформление пояснительной записки	54	24.05.2023 – 13.05.2023	

Дата выдачи задания "26" 04 2023 г.

Руководитель бакалаврской работы _____ Акифьев И.В.
(подпись, дата)

Задание к исполнению принял студент _____ Кочетков К.А.
(подпись, дата)

Работу к защите допустить
Декан факультета _____ Фионова Л.Р.
(подпись, дата)
20.06.2023

Реферат

Выпускная квалификационная работа содержит 70 листов, 32 рисунка, 5 источников, 3 приложения

ИГРОВОЕ ПРИЛОЖЕНИЕ, ГОЛОВОЛОМКА, *UNITY*, *MICROSOFT VISUAL STUDIO 2022*, *C#*

В рамках данной работы была поставлена цель - разработать игровое приложение в жанре «головоломка». Основными требованиями были понятный интерфейс и соответствие правилам данного жанра.

Для достижения этой цели был проведен анализ требований и изучение существующих решений в области разработки игр.

В процессе создания приложения были использованы различные инструменты разработки, такие как язык программирования *C#*, среда разработки *Unity* и интегрированная среда разработки программного обеспечения *Microsoft Visual Studio* 2022.

В результате успешно выполненной работы было создано и отлажено приложение, которое удовлетворяет большинству поставленных задач.

Содержание

Перечень сокращений и обозначений 3

Введение 4

1 Развивающее приложение *Unity* в жанре «Головоломка» 6

1.1 Анализ предметной области и постановка задачи 6

1.1.1 Анализ темы 6

1.1.2 Классификации и виды игр. 7

1.1.3 Жанр разрабатываемой игры 10

1.1.4 Целевая аудитория игр «головоломок» 11

1.1.5 Анализ аналогов 13

1.1.6 Сценарий разрабатываемой игры 15

1.1.7 Постановка задачи 16

1.2 Выбор инструментальных средств 17

1.2.1 Среда разработки *Unity* 17

1.2.2 Язык программирования *C#* 19

1.2.3 Графический редактор *Adobe Photoshop* 20

1.2.4 Редактор *After Effects* 22

1.3 Реализация 23

1.3.1 Интерфейс *Unity* 23

1.3.2 Создание графических элементов при помощи *Photoshop* 29

1.3.3 Создание анимации для персонажа в *After Effects* 31

1.3.4 Создание игрового приложения в *Unity* 34

1.4 Тестирование 42

2 Безопасность жизнедеятельности 46

2.1 Электробезопасность 46
2.2 Требования к освещению рабочего места 47
2.3 Правильное положение за столом 48
Заключение 50
Список использованных источников 51
Приложение А – Листинг программы 52
Приложение Б – *UML* диаграммы 58
Приложение В – Презентация 63

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

ПК – Персональный компьютер

ПО – программное обеспечение

API (*Application Programming Interface*) – интерфейс прикладного программирования, который представляет собой набор определений и протоколов для создания и интеграции прикладного программного обеспечения

AR (*augmented reality* – дополненная реальность) – это технология, которая позволяет через экраны девайсов дополнить реальный физический мир виртуальными изображениями, титрами, анимацией и другими визуальными эффектами

PNG (*portable network graphics*) – растровый формат хранения графической

RPG – Ролевая игра (*role-playing game*) жанр компьютерных игр

информации, использующий сжатие без потерь по алгоритму Deflate

UML (*Unified Modeling Language*) – унифицирован язык моделирования

VR – virtual reality – виртуальная реальность

2D (*Two-dimensional*) – компьютерная двухмерная графика

3D (*Three-dimensional*) – компьютерная трехмерная графика

Введение

Постоянный прогресс характеризует развитие технологий современного мира, где создаются различные устройства, разрабатывается новейшее программное обеспечение. Первоначально разработанные с целью повышения эффективности и решения проблем, технологии в настоящее время все больше ориентируются на индустрию развлечений.

Наблюдается рост разработки развлекательного программного обеспечения, включая видеоигры. За прошедшие годы индустрия видеоигр стала заметным игроком на рынке наряду с кино, анимацией и музыкой. Многие люди используют свои компьютеры и различные устройства для развлечения, а не для работы. В настоящее время большой процент как детей, так и взрослых играет в видеоигры, что приводит к развитию многомиллиардных компаний. Эти компании создают развлекательные продукты и программное обеспечение, состоящие из многочисленных строк кода и мультимедийных файлов.

Доступ к играм расширился во всем мире, и теперь их можно найти на различных игровых устройствах. По мере их развития улучшались такие аспекты, как визуальное пространство, физика объектов и искусственный интеллект. Несмотря на эти достижения, методология разработки осталась неизменной. Количество объектов в виртуальном пространстве ограничено техническими возможностями устройств, для которых предназначена игра. Интерактивность достигается за счет взаимодействия объекта с объектом, объекта со средой и объекта с пользователем, когда информация может быть передана.

Основной проблемой является загрузка кода объектами взаимодействия, отвечающий за выполнение назначенного сценария.

После рассмотрения предоставленной информации было принято решение использовать современные инструменты и методы разработки для создания игрового программного приложения.

В связи с поставленной целью решаются следующие задачи:

– исследовать существующие аналоги программного продукта и выявить их достоинства и недостатки;

– спроектировать алгоритмы для реализации игрового приложения с учетом существующих аналогов;

– реализовать составленные алгоритмы на языке программирования C#;

произвести тестирование представленных алгоритмов.

1 Развивающее приложение *Unity* в жанре «Головоломка»

1.1 Анализ предметной области и постановка задачи

1.1.1 Анализ темы

Программа, созданная для развлечения или обучения, известная как видеоигра (или компьютерная игра, если она используется на компьютере), также может использоваться для целей общения или моделирования сценариев.

Игровые технологии развивались с момента их зарождения в 1950-х годах, и первые устройства в конечном итоге превратились в игровые консоли, портативные консоли и персональные компьютеры, которые сегодня остаются одними из самых популярных игровых платформ.

На сегодняшний день, термин «видеоигры» широко используется для описания всех видов игр, которые требуют использования различных устройств для управления игровым процессом и отображения видеоинформации. Это включает в себя компьютерные игры, игры для портативных устройств, консолей и мобильных устройств.

Видеоигры постоянно развиваются и развиваются по ряду направлений. Вот некоторые из текущих тенденций и разработок в игровой индустрии:

– виртуальная реальность. Произошел всплеск разработки игр виртуальной реальности, которые позволяют игрокам погрузиться в полностью реализованный мир;

– кросс-платформенная игра. Многие игры теперь позволяют игрокам на разных платформах играть вместе, например *Xbox* и ПК или *Playstation* и ПК;

– мобильные игры. Мобильные игры становятся все более популярными, и многие высококачественные игры разрабатываются специально для смартфонов и планшетов;

– киберспорт. Соревновательные игры быстро развиваются, а профессиональные киберспортивные команды и лиги становятся все более распространенным явлением;

– рассказывание историй. Игры все чаще рассматриваются как форма искусства, при этом все большее внимание уделяется рассказу историй и опыту, основанному на повествовании;

– облачные игры. Сервисы облачных игр становятся все более доступными, позволяя игрокам транслировать игры через Интернет без необходимости в дорогостоящем оборудовании.

За прошедшие годы игровая индустрия пережила огромный рост, а разработка компьютерных игр превратилась в гигантскую индустрию. Бюджеты игровых проектов взлетели до небес и в настоящее время достигают десятков миллионов долларов. Рынок современных игр, включая компьютерные, консольные и мобильные игры, значительно вырос и сейчас оценивается в десятки миллиардов долларов. Быстрый рост популярности игр привел к созданию многочисленных вакансий в различных секторах, от игрового дизайна до маркетинга и продаж. Очевидно, что игровая индустрия будет продолжать расширяться, предоставляя бесконечные возможности для тех, кто хочет сделать карьеру в этой области.

1.1.2 Классификации и виды игр.

Игры можно классифицировать несколькими способами, но наиболее распространенная классификация основана на типе игрового процесса. Вот некоторые из наиболее распространенных типов игр:

– экшн-игры. Это игры, требующие быстрой реакции и зрительно-моторной координации. Они часто включают бой, исследование и платформер;

– приключенческие игры. Эти игры сосредоточены на исследованиях и решении головоломок в сюжетной среде. Они часто более медленные, чем экшн-игры;

– ролевые игры (*RPG*). Действие этих игр обычно разворачивается в фэнтезийном мире и включает в себя создание персонажа, повышение его уровня и сюжетные квесты;

– спортивные игры. Эти игры имитируют реальные виды спорта, такие как футбол, баскетбол и футбол;

– стратегические игры. Эти игры требуют тщательного планирования и принятия решений для достижения конкретных целей. Они могут быть пошаговыми или в реальном времени;

– игры-симуляторы. Эти игры имитируют реальный опыт, например полет на самолете, вождение автомобиля или управление городом;

– головоломки. Эти игры заставляют игроков решать сложные головоломки или задачи;

– казуальные игры. Эти игры предназначены для коротких игровых сессий, и в них легко играть;

– образовательные игры. Эти игры предназначены для обучения игроков определенным темам, таким как история, наука или математика;

– многопользовательские игры. Эти игры могут относиться к любому из вышеперечисленных типов, но предполагают игру с другими людьми через Интернет или лично.

По способам реализации игры бывают:

– игры в режиме реального времени;

– конечные и бесконечные;

– со случайными событиями;

– для одного, для нескольких игроков;

– игры с различными уровнями сложности.

По структуре игры делятся на три блока:

– блок правил игры;

– блок анализа игровой ситуации;

– блок взаимодействия с игроком (интерфейс).

Так же можно выделить три основных иерархических уровня, которые позволяют построить схему игры:

– игровой мир. Это высший уровень иерархии, который включает в себя общую игровую среду, в том числе обстановку, в которой происходит игра, правила, регулирующие игровой мир, и цели, которые пытаются достичь игроки;

– игровая механика. Это уровень иерархии, определяющий конкретные действия, которые игроки могут совершать в игровом мире. Он включает в себя такие вещи, как элементы управления, которые игроки используют для перемещения своего персонажа, способности, которые они могут использовать для взаимодействия с окружающей средой и другими игроками, а также правила, регулирующие результаты различных действий;

– игровой контент. Это самый низкий уровень иерархии, включающий все конкретные активы, составляющие игровой мир и определяющие игровую механику. Это включает в себя такие вещи, как модели персонажей и анимация, дизайн уровней, звуковые эффекты и музыка, а также различные визуальные и звуковые эффекты, которые делают игру более захватывающей и приятной для игроков.

Можно классифицировать игры по платформам следующим образом:

– консольные игры. Эти игры предназначены для игр на специальных игровых консолях, таких как *PlayStation, Xbox* и *Nintendo Switch*;

– игры для ПК. Эти игры предназначены для игры на персональных компьютерах (ПК) и часто требуют мощного оборудования и графических карт;

– мобильные игры. Эти игры предназначены для игр на мобильных устройствах, таких как смартфоны и планшеты;

– веб-игры. В эти игры можно играть онлайн с помощью веб-браузера и обычно требуется подключение к Интернету;

– игры виртуальной реальности (*VR*). Эти игры разработаны специально для *VR*-гарнитур и предлагают захватывающий игровой процесс;

– игры с дополненной реальностью (*AR*). В этих играх элементы реального мира сочетаются с цифровым контентом, и в них можно играть с помощью мобильных устройств или специальных технологий *AR*.

Облачные игры. Эти игры транслируются через Интернет, и в них можно играть на различных устройствах, включая ПК, консоли и мобильные устройства.

1.1.3 Жанр разрабатываемой игры

Жанр игр «Головоломки» включает в себя игры, в которых главной задачей игрока является решение различных логических и математических задач. Они могут состоять как из одного уровня, так и из нескольких. В этом мы можем найти разнообразные игры. В этих играх пользователю необходимо пользоваться своими интеллектуальными способностями чтобы продвинуться дальше по сюжету игры. Игры этого жанра могут быть головоломками загадками, головоломками на соображение, головоломками на логику или на способность пользователя быстро принимать нестандартные решения. Головоломки могут быть как настольными, так и компьютерными. Некоторые игры могут быть совсем простыми, например игры для маленьких детей, а другие - очень сложными. Эти игры скорее для любителя чем обычного пользователь.

Одна из причина популярности игр – головоломок – это их доступность. Их можно найти на всех платформах. Компьютерные игры множество из которых не требует мощного компьютера и особых навыков владения им. Телефонные игры особенно популярны потому, что у каждого человека, даже у самых маленьких детей уже есть телефоны. Ещё это очень удобно ведь у каждого телефон под рукой и в любой момент можно зайти в свою любимую игру.

Когда люди играют в головоломки, они учатся не стандартно мыслить, работать в команде.

1.1.4 Целевая аудитория игр «головоломок»

Любители игр головоломок - это люди, которые любят загрузить свой мозг, тем самым повышать свои умственные способности. Эти пользователи как правило разного возраста и разные по сфере деятельности. И вообще их может объединять только желание проверить свои знания и силы.

Студенты и ученики тоже заинтересованы в играх в жанре головоломок. Эти игры помогают им улучшить свои умственные способности в области математики, и логистики.

Программисты и *IT*-специалисты также заинтересованы в решении головоломок. Это та самая профессия где люди, в прямом смысле, каждый день решают сложные логические задачи, а головоломки помогают им прокачать свои навыки.

Многие люди просто хотят провести время с пользой для ума и развлечься одновременно. И игры в данном жанре отлично подходят для этого Они увлекательные, вызывают интерес и создаёт своеобразную интригу.

Игры в жанре головоломок полезны для детей разных возрастов. Когда дети играют, они сосредоточиваются на задаче и не отвлекаться на другие вещи тем самым тренируют внимание и концентрацию.

Многие головоломки связаны с математическими вычислениями и арифметическими операциями. При решении таких задач у детей развиваются математические навыки.

Решение сложных головоломок может очень удовлетворить ребенка, что повышает их самооценку и уверенность в своих способностях.

Различные типы головоломок могут более или менее подходить для детей разного возраста. Например, простые головоломки с картинками где необходимой найти какой-то предмет больше подходят для маленьких детей,

а сложные головоломки на логику и математические задачи - для более взрослых детей. Также необходимо выбирать игры, которые соответствуют интересам конкретного ребенка, чтобы игры были не только полезными, но и интересны ребенку.

Так же необходимо учитывать, что головоломки, да и в целом любые игры, могут негативно воздействовать на ребенка. Они могут вызывать стресс. Некоторые загадки могут быть очень сложными, из-за чего ребёнок может распсиховаться, стать нервным. Поэтому, необходимо следить за тем, как часто и как долго дети играют в головоломки. Хотя некоторые головоломки можно решать вместе, это может и не поспособствовать развитию коммуникативных и социальных навыков также эффективно, как взаимодействие с реальными людьми. Хотя игры в жанре головоломок могут быть полезны для развития когнитивных и математических навыков, они не обучают другим, таким как физическая активность, способность жить в социуме, и т.д. Риск замещения более важных видов обучения. Если дети проводят слишком много времени в играх, это может привести к игнорированию развития более важных видов обучения, таких как чтение, учеба и другие формы развития.

Конечно, все эти минусы не являются обязательными и зависят от конкретной ситуации. Важно следить за тем, чтобы игры в жанре головоломок не были единственным видом развития для ребенка, и чтобы использовались в разумных количествах и в соответствии с возрастом и уровнем развития каждого ребенка.

Головоломки могут иметь большое положительное влияние на пожилых людей, включая следующие аспекты:

Улучшение когнитивных функций: Игры головоломки могут помочь улучшить память, концентрацию, логическое мышление и другие когнитивные функции. У пожилых людей часто возникают проблемы с

памятью и мышлением, поэтому регулярное решение головоломок может помочь поддерживать эти функции на должном уровне.

Стимулирование мозга: Решение сложных головоломок требует от мозга дополнительного напряжения и стимуляции, что может быть полезно для его здоровья в целом. Более того, игры головоломки могут помочь улучшить связи между нейронами и сохранить общую работоспособность мозга.

Социальные выгоды: Игры головоломки могут предоставить возможность для социального взаимодействия. Пожилые люди могут решать головоломки вместе с другими или обсуждать свои решения. Это может помочь снизить социальную изоляцию и улучшить качество жизни.

Снижение стресса: Решение головоломок может помочь пожилым людям расслабиться и снять стресс. Это может быть особенно полезно для тех, кто страдает от беспокойства или тревоги.

Улучшение настроения: Решение головоломок может поднять настроение и дать ощущение достижения. Это может быть особенно важным для пожилых людей, которые могут чувствовать себя усталыми или подавленными из-за проблем со здоровьем или других факторов.

Головоломки могут быть очень полезны для пожилых людей, как с точки зрения улучшения когнитивных функций и здоровья мозга, так и для социального взаимодействия, снятия стресса и улучшения настроения.

1.1.5 Анализ аналогов

При разработке игры необходимо учесть возможности существующего на данный момент ПО для повышения конкурентоспособности программного продукта.

1.1.5.1 Игра - *Monument Valley*

Monument Valley - это игра-головоломка, разработанная студией *Ustwo Games* и выпущенная в 2014 году. В игре вы играете за принцессу Иду,

которая путешествует по фантастическому миру, решая головоломки, чтобы достичь конца каждого уровня.

Основным элементом *Monument Valley* являются его невероятные архитектурные конструкции, которые кажутся необъяснимыми. Вы будете перемещать фрагменты ландшафта, чтобы создавать пути для Иды, взаимодействовать с механизмами и использовать оптические иллюзии для прохождения каждого уровня.

Кроме того, игра имеет потрясающую визуальную составляющую, с великолепной графикой и уникальным звуковым дизайном. *Monument Valley* - это не только отличная игра-головоломка, но и настоящее произведение искусства.

1.1.5.2 Игра *DOP* 2

DOP 2 - это игра-головоломка с элементами приключения, созданная компанией *SayGames*. В этой игре вы играете за детектива, которому нужно разгадать ряд таинственных дел, используя свои наблюдательные способности и логическое мышление.

Каждый уровень в *DOP* 2 представляет собой небольшую историю, которая содержит множество подсказок и скрытых объектов. Игрокам нужно изучать каждый уровень, общаться с персонажами, выполнять задания и решать головоломки, чтобы продвигаться по сюжету.

Головоломки в *DOP* 2 могут быть очень разнообразными: от расшифровки кодов до поиска спрятанных предметов на экране. Если вы застряли на каком-то уровне, есть возможность использовать подсказки или пропустить его, но это потребует определенного количества монет.

В целом, *DOP* 2 - это увлекательная и захватывающая игра для тех, кто любит разгадывать головоломки и искать скрытые предметы. Она также имеет яркую графику и забавный звуковой дизайн, что делает игровой процесс еще более интересным.

1.1.5.3 Игра *Words of Wonders*

Words of Wonders - это игра-головоломка для мобильных устройств, в которой необходимо составлять слова из букв, расположенных на поле. Цель игры - найти все слова, которые можно составить из заданных букв.

Игра содержит множество уровней, каждый из которых представляет собой картинку с достопримечательностью или красивым пейзажем, а также список слов, которые нужно найти. Пользователи могут перетаскивать буквы, чтобы составить слово, и получать бонусные очки за нахождение слов, которые не указаны в списке.

Кроме того, игра предоставляет подсказки в случае затруднения. Например, пользователь может использовать кнопку «подсказка», чтобы узнать первую букву слова. Если это не помогает, можно запросить еще одну подсказку, но за это будет списываться часть очков.

Игра *Words of Wonders* подходит для людей всех возрастов, поскольку она не требует особых знаний или навыков, а также помогает развивать лексический запас и решительность.

1.1.6 Сценарий разрабатываемой игры

Нас встречает подводный мир с парой давно затонувших кораблей. Внезапно из-за скалы появляется маленький розовый осьминог с милой улыбкой и ползет по дну океана.

Познакомьтесь с морским гением, маленьким розовым осьминогом, который живет на дне океана. Он любит исследовать свой подводный мир и решать головоломки. Сегодня перед ним стоит задача собрать все ключи на локации и найти сокровище.

Игра представляет собой головоломку, в которой игрок управляет «Морским гением», и помогает ему решать головоломки, собирать ключи и находить сокровища. Игра ведется с боковой прокруткой, когда осьминог ползет по дну океана.

Герой должен преодолевать различные препятствия, такие как камни и парящие на пустых бутылках скалы, чтобы добраться до ключей. Ключи разбросаны по всей локации, и игрок должен придумать, как до них добраться, расставив предметы на локации. Среди предметов есть камни, пустые бутылки и другие объекты, которые можно перемещать.

По мере того, как игрок продвигается по уровням, головоломки становятся все более сложными, и осьминожек должен использовать свои навыки решения проблем, чтобы добраться до ключей и найти сокровища.

Собрав все ключи и открыв проход к сокровищам, осьминог проползает через узкий проход и появляется в большой пещере. В центре пещеры он видит большой сундук с сокровищами. Окто открывает сундук, и он наполнен золотом и другими сокровищами.

1.1.7 Постановка задачи

Необходимо разработать игровое приложение на *Unity*. Приложение должно удовлетворять следующим требованиям:

– понятный игровой интерфейс ;

– красивая и яркая графика;

– разнообразные объекты, которые персонаж может передвигать и устанавливать в необходимые места для того, чтобы продвинуться дальше по уровню;

– управление в приложении должно работать правильно и быть отзывчивым;

– реализовать качественные анимации для персонажа;

– реализовать интерфейс пользователя, который включает в себя главное меню, внутриигровой интерфейс с инвентарём и кнопкой перезапуска сцены.

На рисунке 1 представлена диаграмма использования меню.

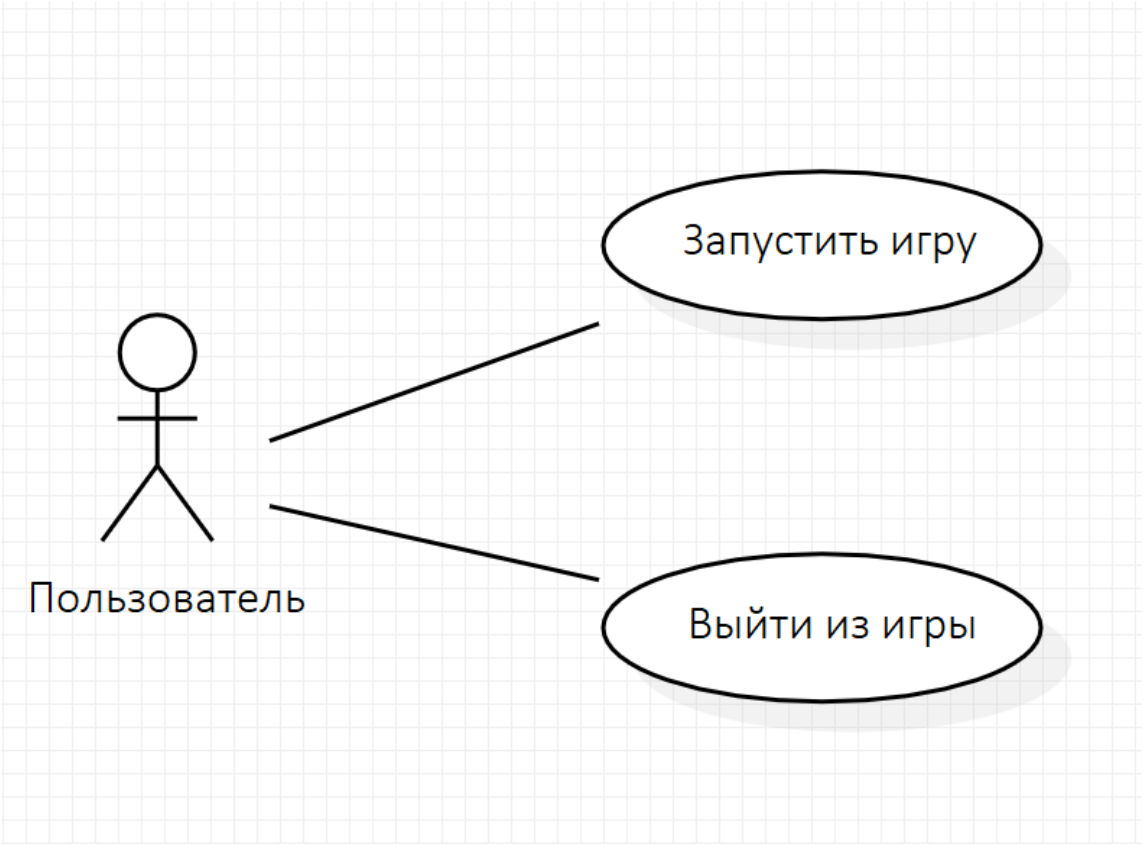


Рисунок 1 – Диаграмма использования стартового меню

1.2 Выбор инструментальных средств

1.2.1 Среда разработки *Unity*

Unity - это кроссплатформенная среда разработки программного обеспечения, специализированная на создании игр и визуализации 3D-моделей. *Unity* позволяет разрабатывать игры для различных платформ, включая компьютеры, мобильные устройства, консоли и веб-браузеры.

Unity имеет мощный графический движок, который поддерживает стандарты графики, такие как *DirectX* и *OpenGL*. Это позволяет создавать высококачественную графику и реалистичные эффекты в играх. *Unity* также имеет инструменты для работы с 3*D*-моделями, анимацией персонажей и объектов, и поддерживает скрипты на нескольких языках программирования, включая *C#, JavaScript* и *Boo*. Главное окно программы представлено на рисунке 2.

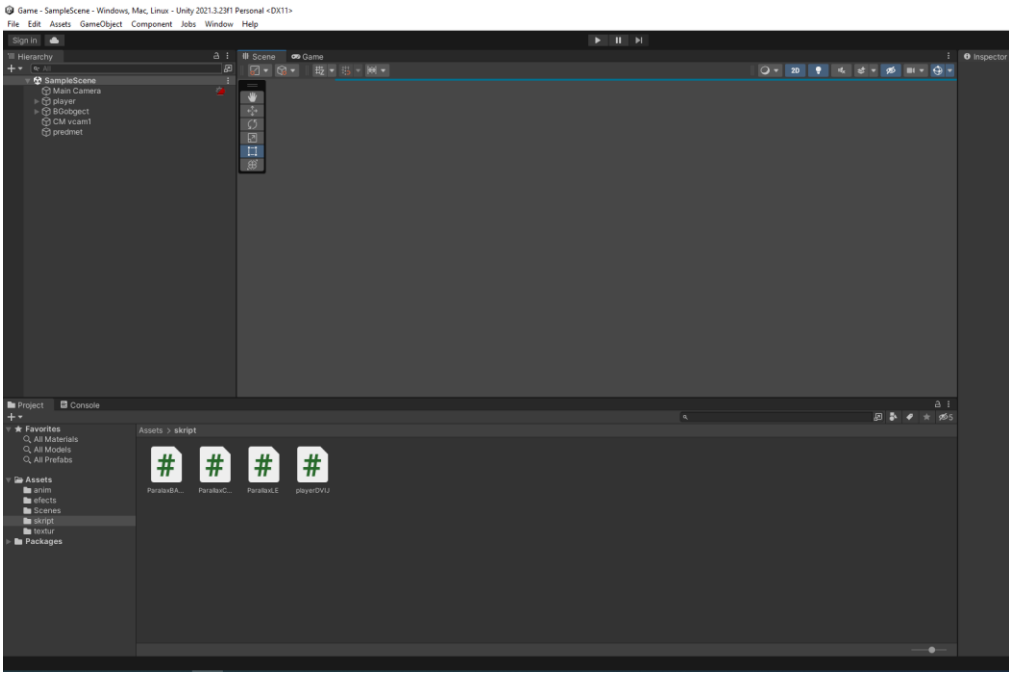


Рисунок 2 – Меню *Unity*

Кроме того, *Unity* имеет большое сообщество разработчиков, которые делятся своими знаниями и создают готовые ресурсы для использования в играх, такие как модели персонажей, текстуры, звуковые эффекты и другие активы. В *Unity* также есть встроенный магазин активов, где можно найти и приобрести нужные ресурсы. Этот движок позволяет создать сцену, на которой будут размещены импортированные из 3*D's Max* объекты взаимодействия.

Особенности:

– кроссплатформенность. *Unity* позволяет создавать игры для разных платформ, используя один и тот же код;

– графика. *Unity* имеет мощный графический движок, который поддерживает стандарты графики, такие как *DirectX* и *OpenGL*;

– легкоизменяемый пользовательский интерфейс. *Unity* предоставляет простой и интуитивно понятный пользовательский интерфейс, который может быть легко настроен и изменен;

– широкие возможности по созданию анимации. *Unity* имеет много инструментов для создания и управления анимацией персонажей и объектов;

– возможность работы с 3*D*-моделями. *Unity* имеет встроенный инструмент для работы с 3*D*-моделями, что делает его очень удобным для создания трехмерных игр;

– сообщество. *Unity* имеет большое и активное сообщество разработчиков, которые делятся своими знаниями и помогают друг другу;

– *asset Store*. *Unity* имеет встроенный магазин активов, где можно найти и приобрести готовые модели, текстуры, звуковые эффекты и другие ресурсы для использования в играх;

– поддержка скриптов на разных языках. *Unity* поддерживает скрипты на *C#, JavaScript* и *Boo,* что дает разработчикам больше свободы в выборе языка программирования;

– легкоизменяемая физика. *Unity* имеет встроенный физический движок, который может быть легко настроен и изменен для создания различных физических эффектов в играх.

Unity, несомненно, является одним из самых удобных и мощных игровых движков, доступных на сегодняшний день. Его универсальность и способность работать на нескольких платформах делают его идеальным выбором для многих разработчиков. Решение об использовании именно этой многоплатформенной среды разработки было принято после тщательного изучения ее особенностей и преимуществ.

1.2.2 Язык программирования *C#*

В качестве первого ключевого языка программирования в данной дипломной работе выбран *C#*. Пространство языка программирования *C#* представлено на рисунке 3.

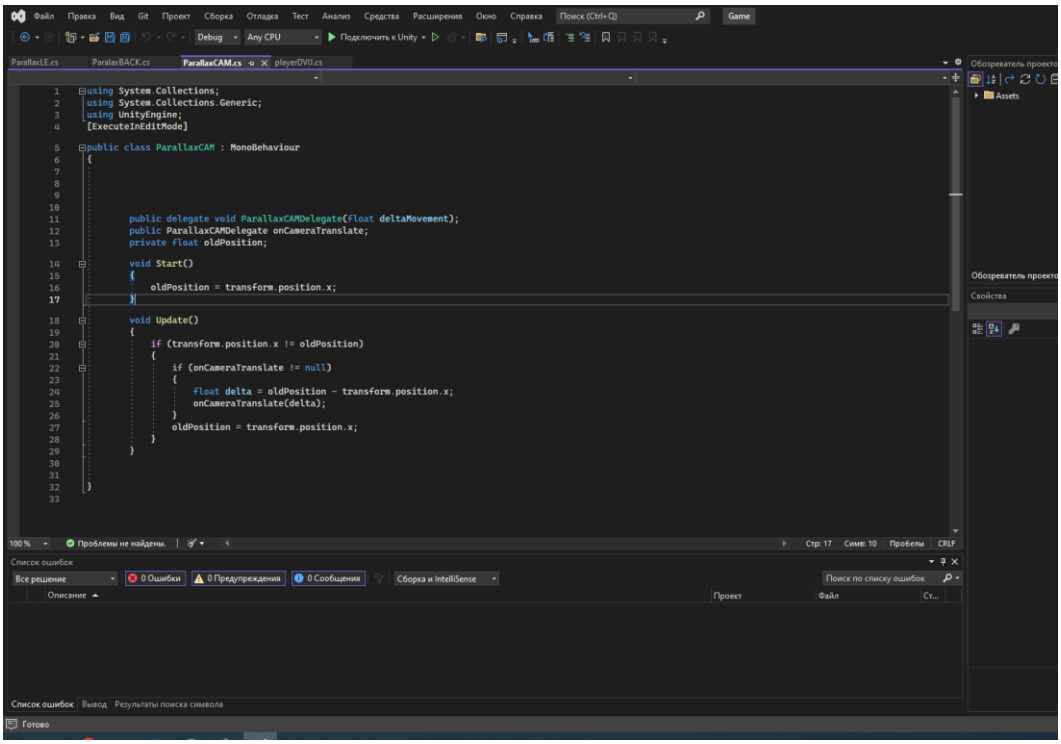


Рисунок 3 – Пространство языка *C#*

C# — это язык программирования, завоевавший популярность среди разработчиков благодаря своему синтаксису, похожему на *C*. Хотя синтаксис *C#* больше похож на *C++* и *Java*, он также многое унаследовал от других языков, таких как *Delphi*, *Modula* и *Smalltalk*. *C#* был разработан, чтобы исключить проблемные модели разработки программного обеспечения, которые присутствовали в его предшественниках. Например, в отличие от *C++*, *C#* не допускает множественного наследования классов, хотя допускает множественное наследование интерфейсов.

Благодаря своим полезным функциям *C#* стал основным языком для многих разработчиков. Он включает встроенную поддержку шаблонов, делегатов и событий, что упрощает его реализацию. Кроме того, *C#* — отличный язык для разработки кроссплатформенных приложений, что делает его лучшим выбором для разработчиков по всему миру.

1.2.3 Графический редактор *Adobe Photoshop*.

Photoshop - это программа для работы с графическими изображениями, которая позволяет редактировать и создавать различные цифровые

произведения. В контексте разработки игр в *Unity*, *Photoshop* может использоваться для создания спрайтов, текстур, анимаций персонажей и других элементов графики. Интерфейс редактора представлен на рисунке 4.

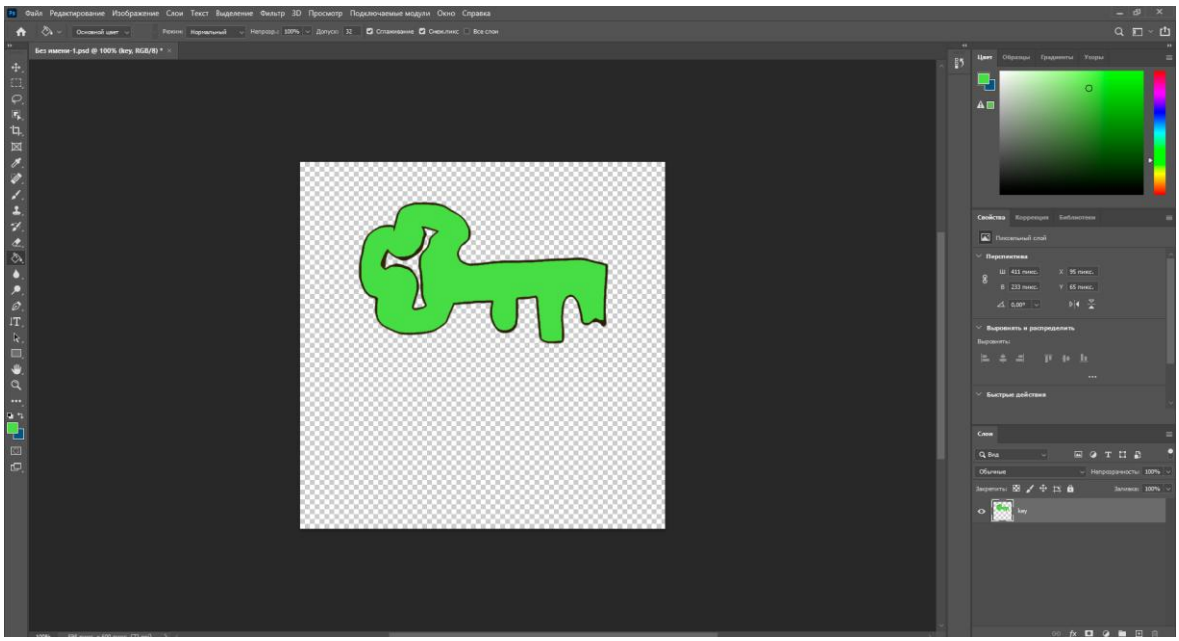


Рисунок 4 – Пространство редактора *Adobe Photoshop*

Для создания спрайтов мы можем использовать инструменты *Photoshop*, такие как кисти и заливка, чтобы создать детальную иллюстрацию персонажа или объекта. Затем мы можем экспортировать наше изображение как *PNG*-файл и импортировать его в *Unity* как спрайт.

Photoshop также может использоваться для создания текстур, которые могут быть применены к поверхностям объектов в нашей игре. Например, мы можем создать текстуру для заднего фона или пола(песка), используя инструменты *Photoshop*, такие как градиент и фильтры, затем экспортировать эту текстуру в формате *PNG* и импортировать ее в *Unity*.

Кроме того, *Photoshop* можно использовать для создания интерфейсов пользовательских приложений, логотипов и многих других графических элементов, которые могут быть использованы в *2D* играх в *Unity*.

1.2.4 Редактор *After Effects*

Adobe After Effects - это программа для создания и редактирования профессиональной анимации, видеоэффектов, специальных эффектов, композитинга и многого другого. Она используется в качестве инструмента для создания движущейся графики, которая может быть использована в рекламных роликах, фильмах, телевизионных передачах, веб-контенте и многих других мультимедийных проектах.

С помощью *Adobe After Effects* мы можем создавать различные анимации, например, 2*D* создавать текстовые эффекты, добавлять переходы, оптические и другие спецэффекты, создавать титры и многое другое. Функциональность программы позволяет создавать как простые, так и сложные анимации с учетом всех деталей.

В разработке данного игрового приложения *Adobe After Effects* будет применятся для создания анимации нарисованным в Photoshop персонажам. Рабочее пространство *After Effects* представлено на рисунке 5.

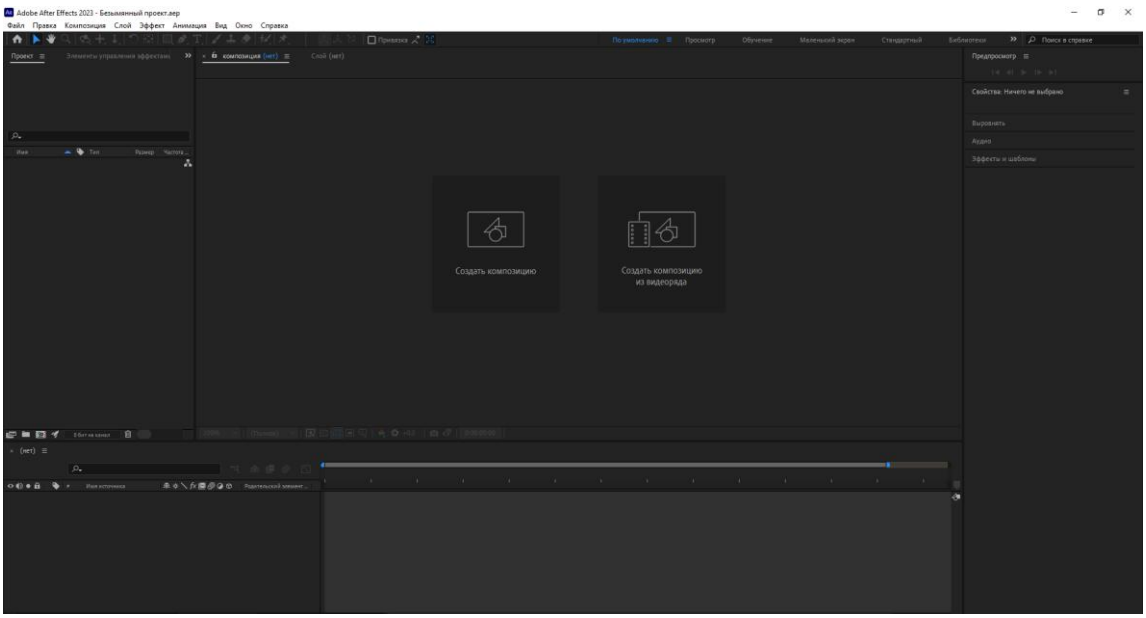


Рисунок 5 – Рабочее пространство *After Effects*

1.3 Реализация

1.3.1 Интерфейс *Unity*

Сцена (англ. "*Scene*") в *Unity* - это пространство, где размещаются игровые объекты и настраивается среда игры. Она представляет собой композицию объектов, которые рендерятся на экране во время игры.

В сцене можно добавлять и удалять игровые объекты, настраивать их свойства, располагать их относительно друг друга или окружения, а также создавать и настраивать освещение и другие эффекты.

Сцены в *Unity* используются для организации игрового пространства на уровне проекта. Когда вы создаете новый проект в *Unity*, система автоматически создает для вас стандартную сцену. Мы можем создавать новые сцены и переключаться между ними, чтобы работать над различными частями проекта.

Кроме того, сцены могут быть использованы для создания уровней игры. Вы можете создать несколько сцен - для каждого уровня - и переключаться между ними при прохождении игроком игры. Это позволяет создавать более сложные игровые ситуации, включая различные локации, задания, условия победы и т.д.

Использование сцен в *Unity* позволяет добиться более организованного и эффективного процесса разработки игры, упрощает работу с компонентами и настройкой окружения. Сцена представлена на рисунке 6.

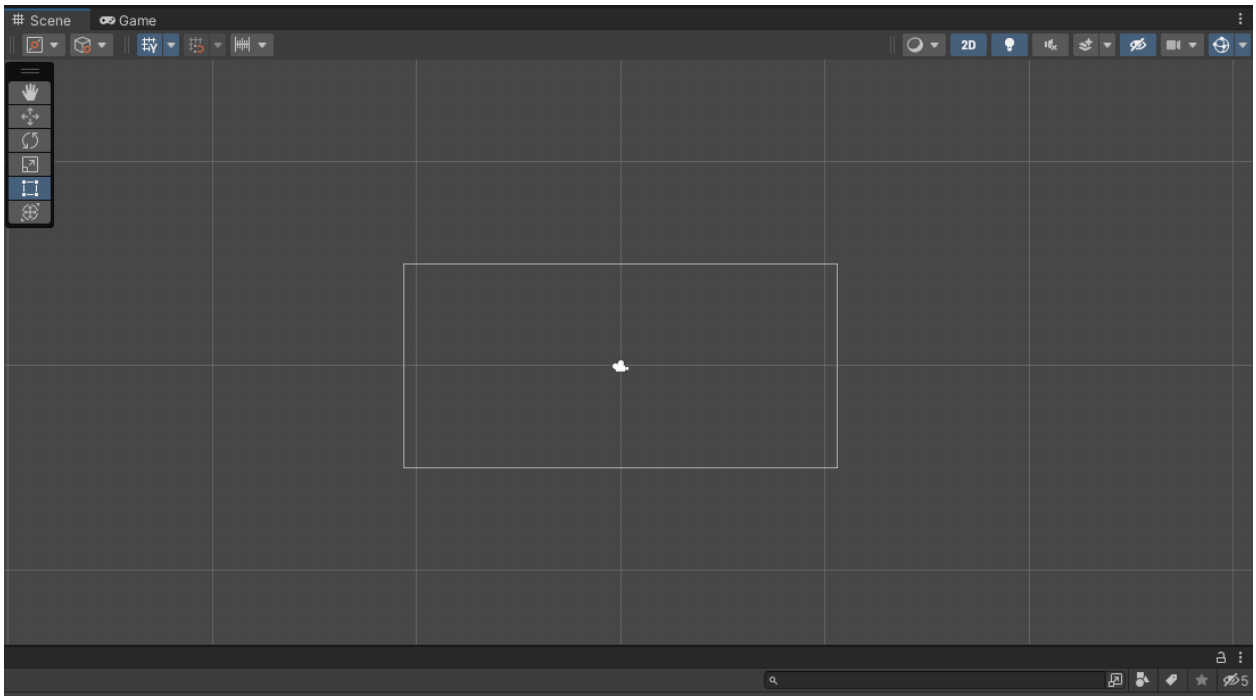


Рисунок 6 – Сцена проекта

Иерархия (hierarchy) - это окно, в котором отображаются объекты сцены в виде древовидной структуры. Она показывает, как объекты связаны друг с другом и как они расположены на сцене.

Когда мы создаем новый объект в *Unity*, он по умолчанию помещается в корень иерархии. Мы можем перемещать его в другие объекты или создавать дочерние объекты, чтобы организовать их логически и удобно управлять ими.

Использование иерархии облегчает работу с объектами на сцене. Например, мы можем быстро выбирать объекты, менять их параметры и добавлять компоненты, а также создавать скрипты, которые будут взаимодействовать с объектами в иерархии. Окно иерархии представлено на рисунке 7.

Рисунок 7 – Иерархия проекта

Окно *Game* (игровое окно) в *Unity* - место, где мы можем видеть и работать со своей игрой. Окно *game* отображает, что увидит пользователь или мы когда запустим игру.

В окне *Game* вид с камеры, которую вы добавили на сцене. Мы можем перемещать камеру, изменять её параметры, можем расставить объекты взаимодействия на сцене и смотреть в этом окне, как они друг на друга влияют.

Окно *Game* даёт представление, как игра будет выглядеть для пользователя. Всё это помогает нам быстро понять свою ошибку, наглядно увидеть в чём она заключается и быстро исправить ее. Окно *Game* представлено на рисунке 8.

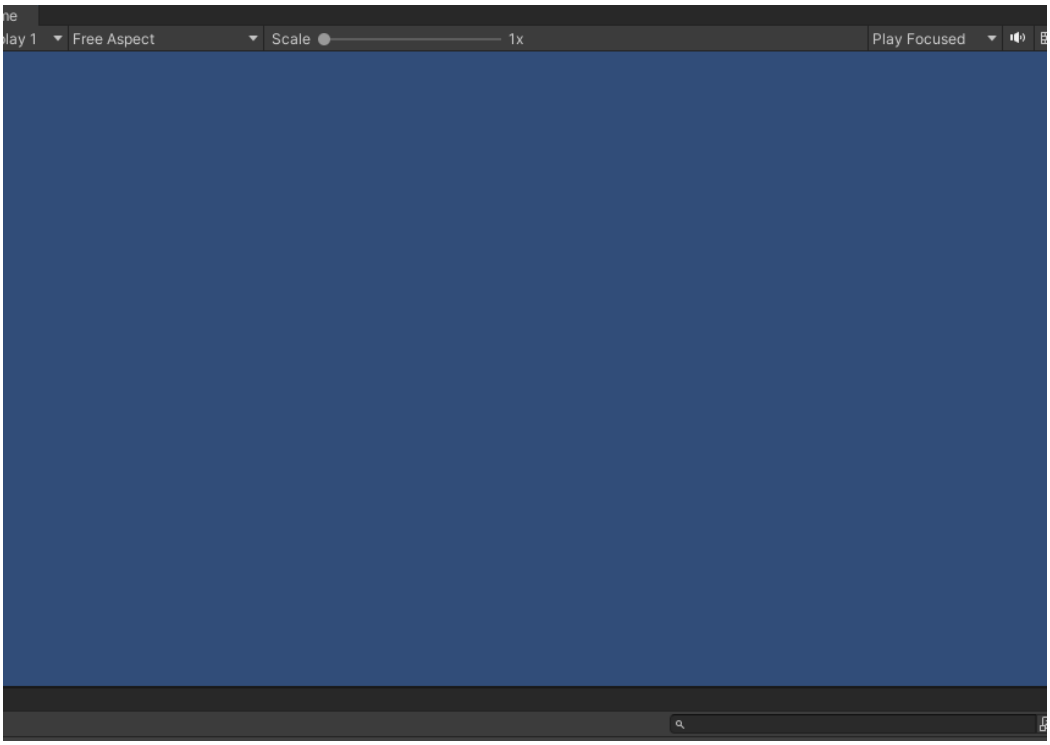


Рисунок 8 – Окно *Game*

Inspector – это окно, которое показывает всю информацию о выбранном объекте. Это окно дает доступ к свойствам, компонентам и параметрам объекта, что позволяет быстро изменять их.

Когда мы выбираем объект на сцене, его параметры отображаются в окне *Inspector*. Мы можем легко изменять их. Например, если мы выберем объект, мы сможем изменить его расположение, поворот, масштаб и т.д.

Окно *Inspector* также помогает добавлять компоненты к объекту и изменять их. Мы можем их выбрать из огромного множества. К ним можно отнести коллайдеры, скрипты, аниматоры и т.д. Каждый компонент имеет свои параметры, которые можно настроить в окне *Inspector*.

Окно инспектора представлено на рисунке 9.

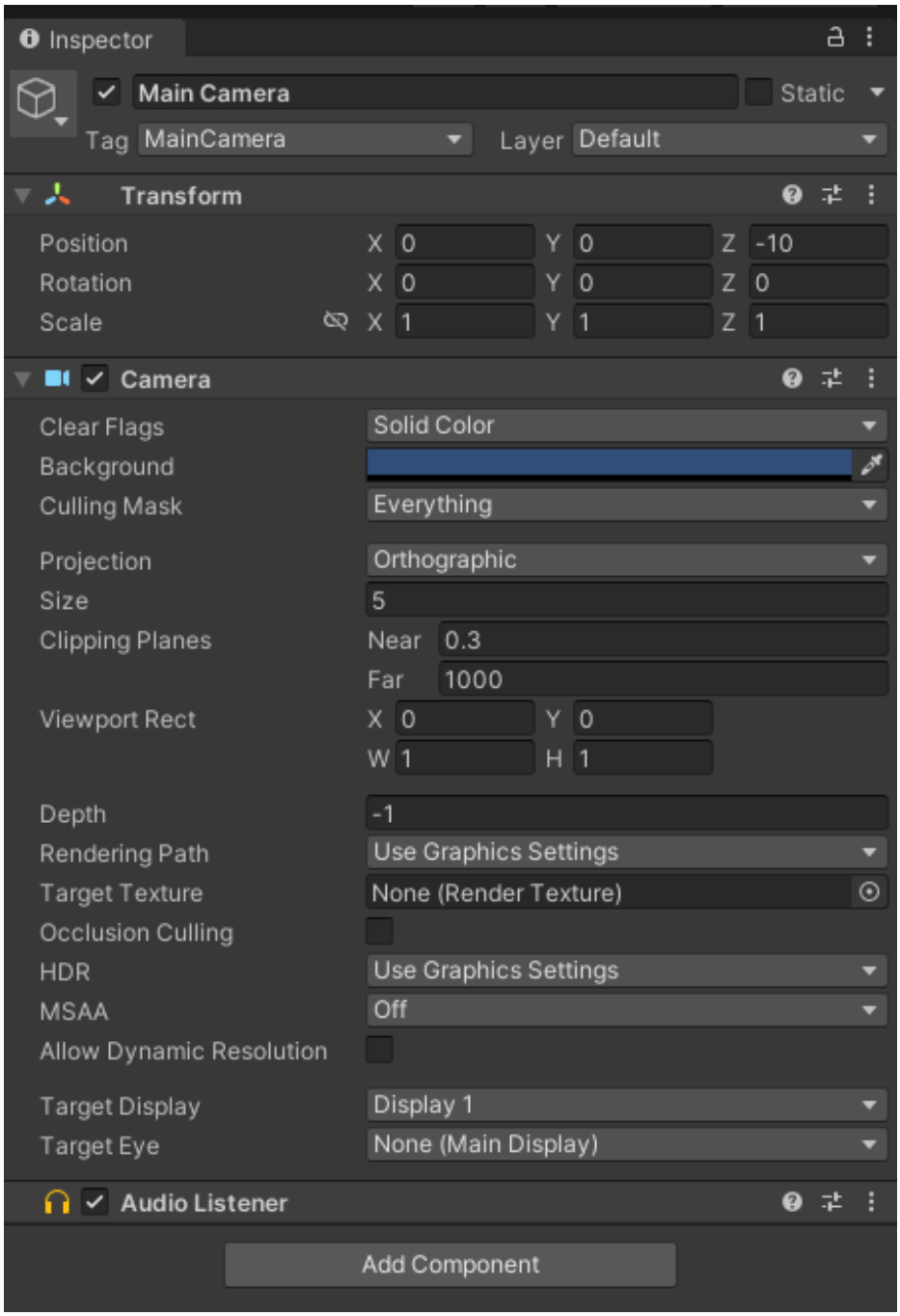


Рисунок 9 – Окно *inspector*

Окно *Project* в *Unity*. Это окно, которое отображает все файлы и ресурсы проекта, такие как, сцены, рисунки персонажей, скрипты, звуковые эффекты.

В окне *Project* можно создавать новые файлы и папки, перемещать, копировать и удалять файлы. Оно удобно для быстрого поиска необходимого скрипта, звука или картинки.

Еще одна важная функция окна *Project* - это отображение структуры проекта в виде дерева. Мы можем расставить папки и файлы так, как удобно любому разработчику, когда все папки и файлы в проекте раскиданы в нужном порядке, тогда и ускоряется процесс разработки игр.

В окне *Project* также можно быстро выбрать необходимый объект и изменить его свойства. Всё это сильно облегчает разработку игровых приложений. Окно project изображено на рисунке 10.

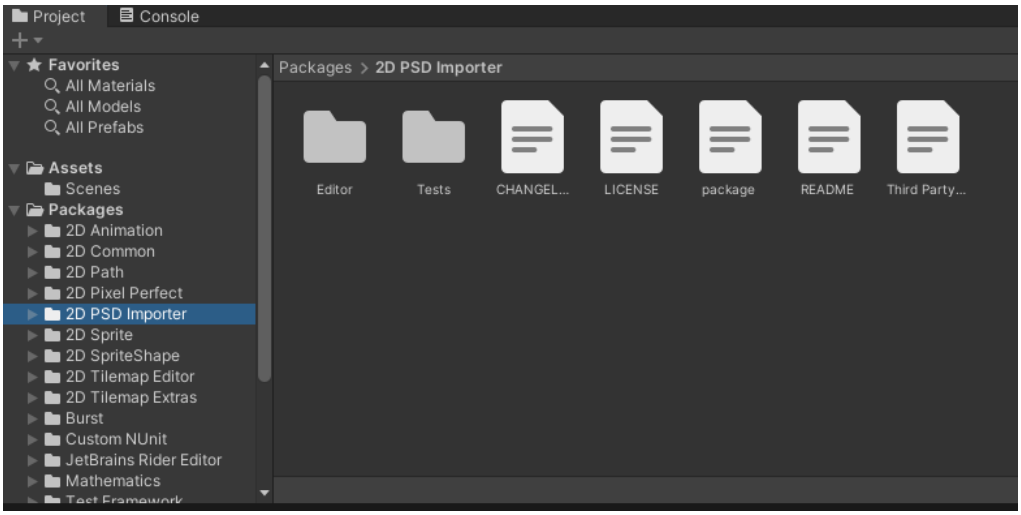


Рисунок 10 – Окно *project*

Console - это окно, в котором отображается информация о работе игры и любые ошибки и предупреждения, которые могут возникнуть при ее запуске.

После запуска игры, все сообщения, которые выводят наши скрипты и движкок *Unity*, будут отображаться в консоли. Это помогает быстро обнаружить и исправить ошибку.

Ещё, когда мы импортируем различные ресурсы консоль показывает сообщения об успешном выполнении и добавлении компонентов.

Окно *Console* позволяет фильтровать сообщения по типу (ошибки, предупреждения, сообщения). Это может быть полезно при разработке и отладке более крупных проектов.

Консоль это важный инструмент для разработки игр в *Unity*. Он помогает обнаруживать и устранять ошибки и проблемы. Окно консоли представлено на рисунке 11.

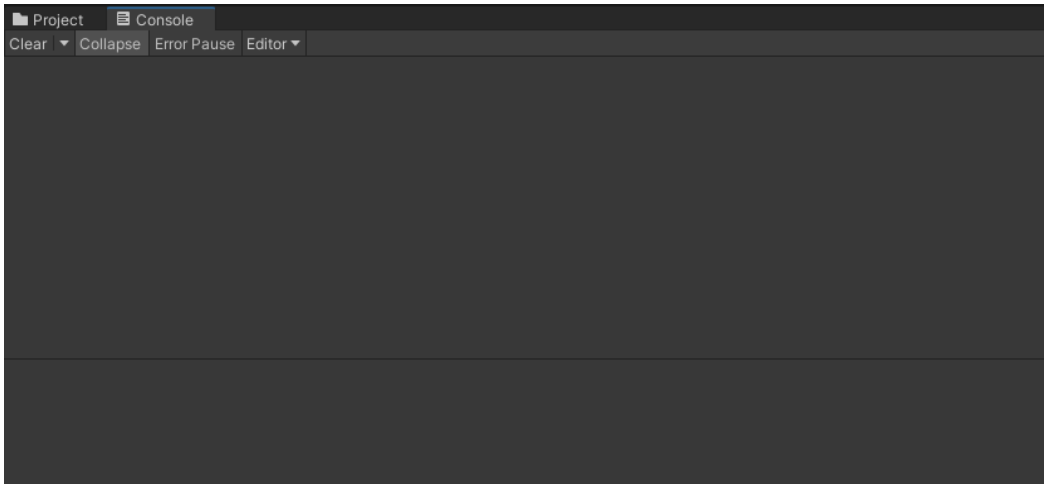


Рисунок 11 – Окно *console*

1.3.2 Создание графических элементов при помощи *Photoshop*

Для начала нужно нарисовать основную сцену в несколько слоёв. Это понадобится в дальнейшем для создания параллакс эффекта. Параллакс-эффект это визуальный эффект, при котором объекты на заднем плане движутся медленнее, чем объекты на переднем плане, создавая ощущение глубины и реалистичности окружающего мира. Вид созданной сцена представлен на рисунке 12.

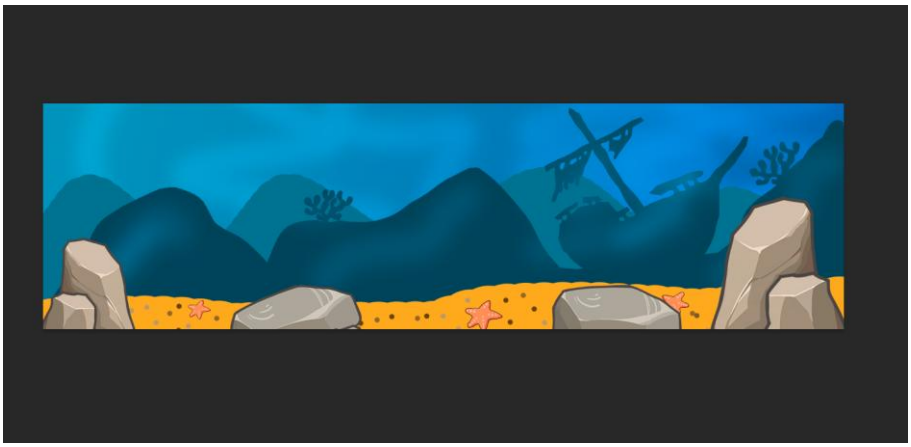


Рисунок 12 – Сцена игры

Теперь необходимо нарисовать главного персонажа «морского гения». Осьминог (главный герой) представлен на рисунке 13.

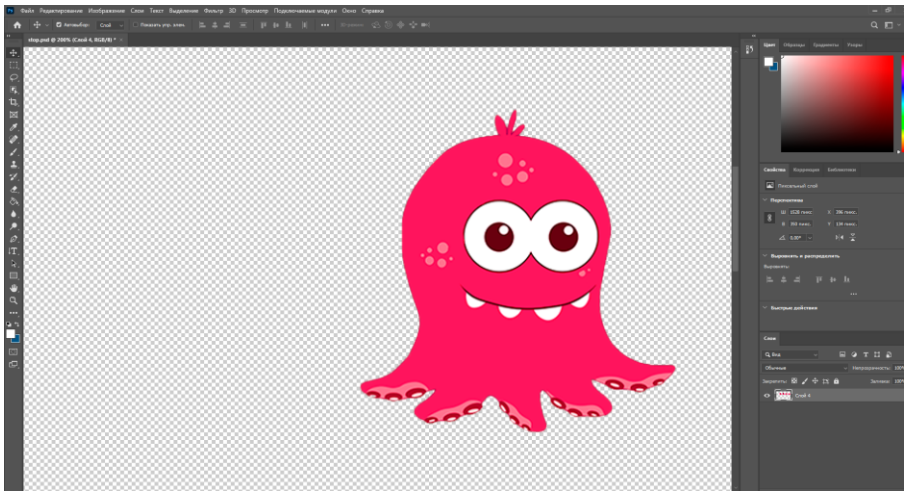


Рисунок 13 – Главный персонаж

По сценарию игры персонаж должен собирать ключи чтобы открыть путь к сокровищу, необходимо нарисовать ключи и само сокровище, представленное на рисунке 14.

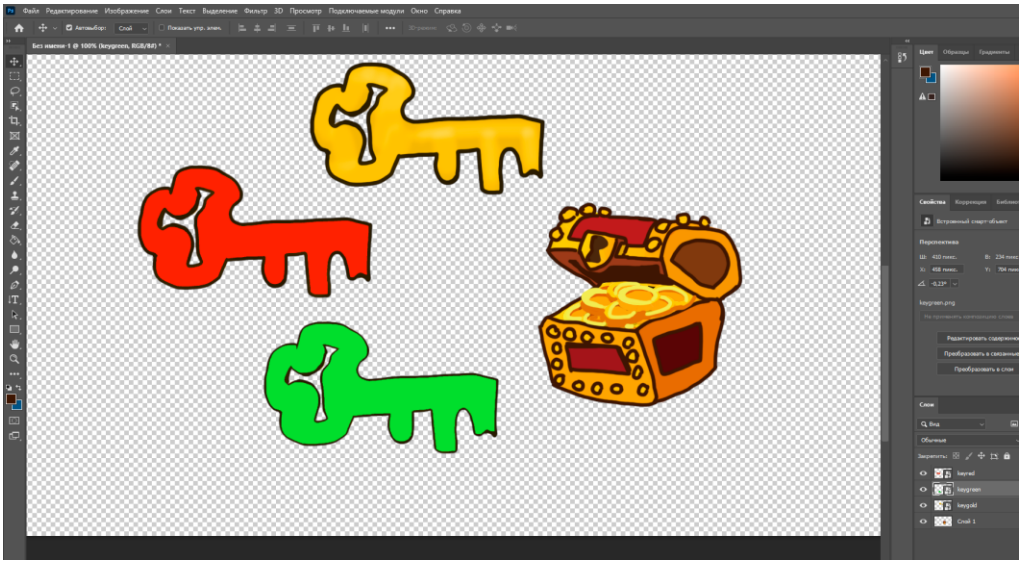


Рисунок 14 – Предметы взаимодействия

Так же для игры необходимо нарисовать меню и предметы взаимодействия. Они представлены на рисунке 15, 16.

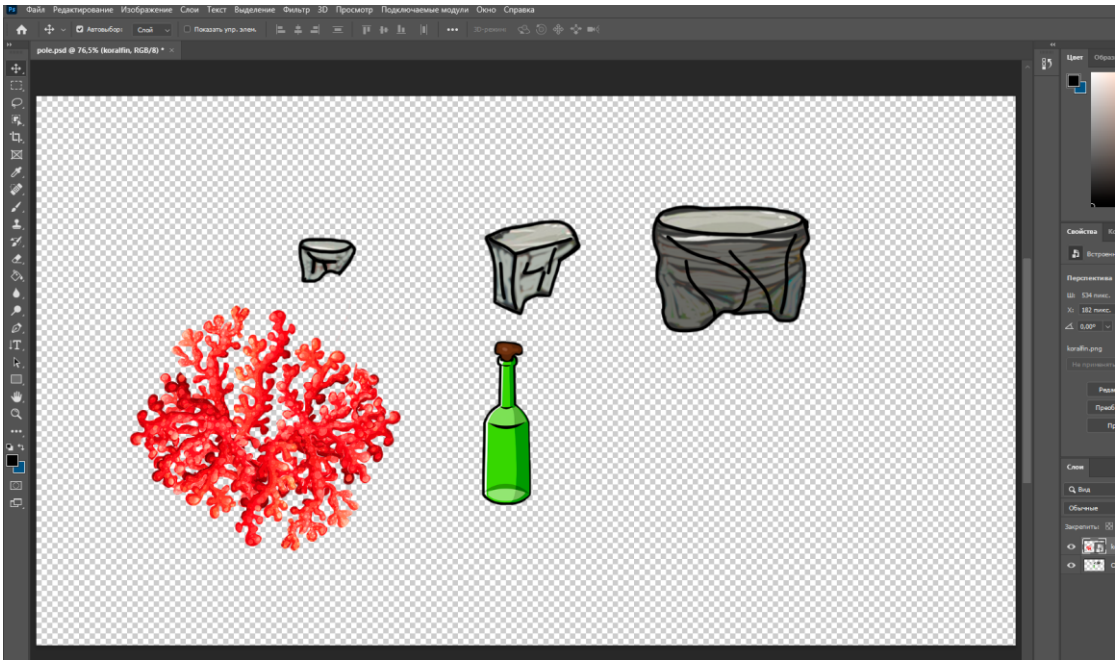


Рисунок 15 – Предметы взаимодействия

Рисунок 16 – Кнопки для меню

1.3.3 Создание анимации для персонажа в *After Effects*

Для анимации *PNG* картинки при помощи *After Effects* необходимо воспользоваться инструментом «булавка положения марионетки».

Инструмент "Булавка" (*Pin Tool*) в *After Effects* используется для закрепления (привязки) конкретной точки на слое к определенной позиции

на другом слое или объекте. Это особенно полезно при работе с анимацией персонажей, когда необходимо точно управлять перемещением отдельных частей тела.

Чтобы использовать инструмент "Булавка", необходимо выбрать его и затем щелкнуть на слое, который необходимо закрепить. После этого вы можно переместить точку привязки на нужную позицию на другом слое или объекте. Когда двигается первый слой, точка привязки будет следовать за ним, сохраняя свое положение на другом объекте.

Использование инструмента "Булавка" помогает создавать более реалистичные анимации, такие как движение конечностей персонажа или механических деталей. Он также может быть использован для создания эффекта сцепления объектов, когда один объект должен точно следовать за другим во время движения. Расставленные точки и пример работы инструмента представлены на рисунке 17.

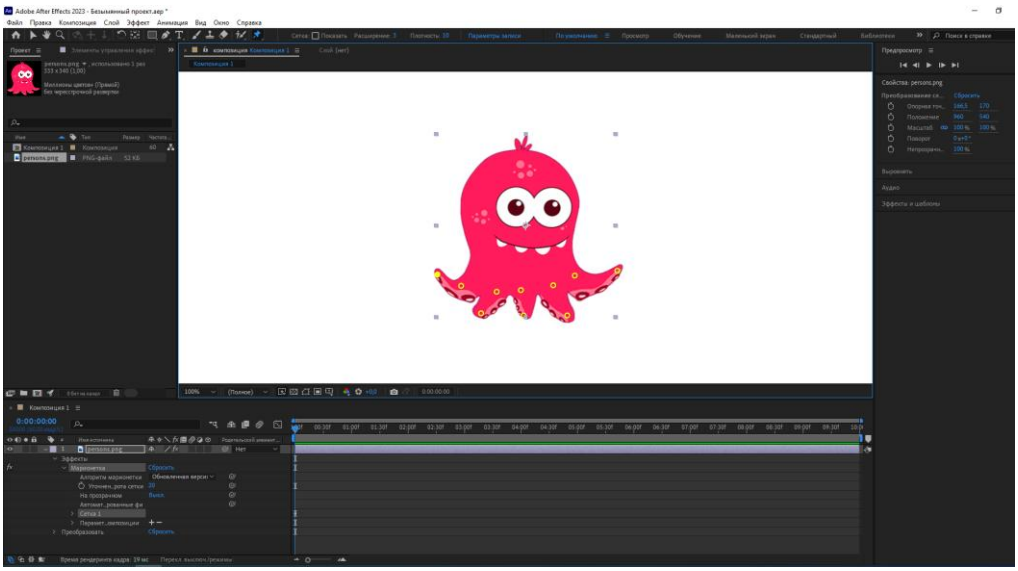


Рисунок 17 – Работа с инструментом булавка

После создания анимации в редакторе *After Effects* необходимо разделить анимацию на 8 частей и вырезать персонажа при помощи *Photoshop*. Результат представлен на рисунке 18–20.

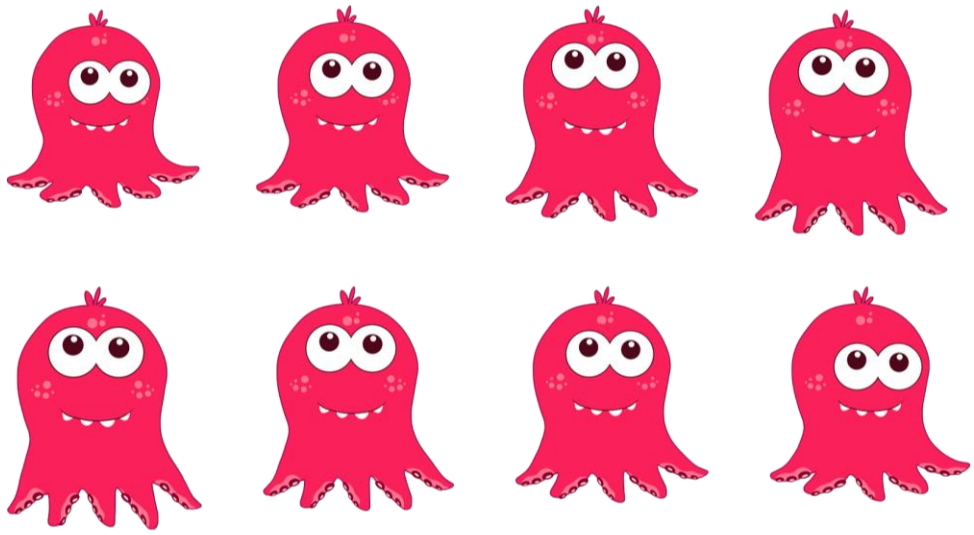


Рисунок 18 – Спрайтовая анимация прыжка

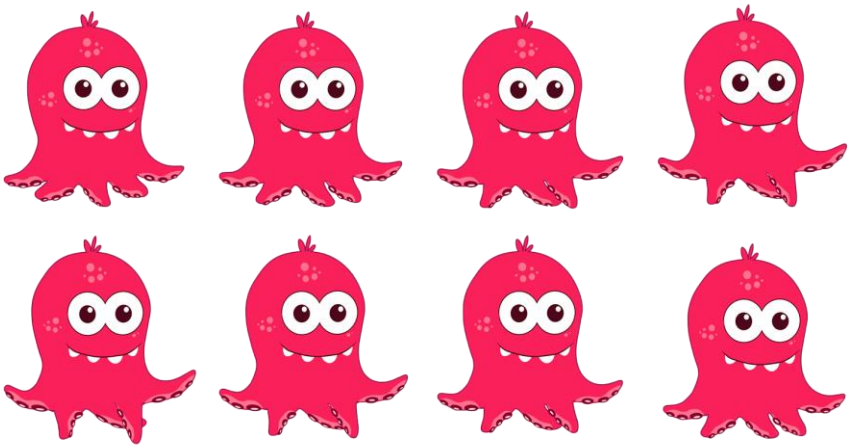


Рисунок 19 – Спрайтовая анимация движения

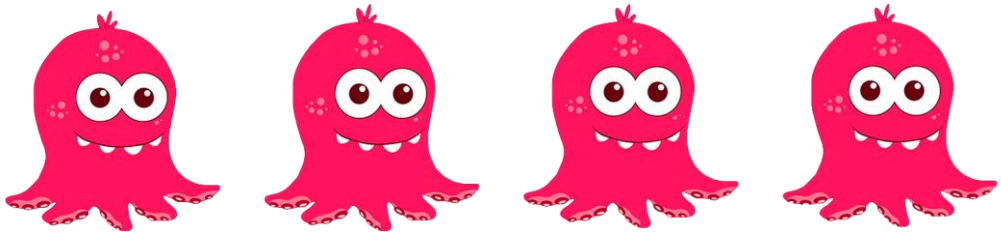


Рисунок 20 – Анимация небольшого движения глаз

1.3.4 Создание игрового приложения в *Unity*

Unity 2 вида анимации, которые позволяют создавать динамические игровые объекты. Давайте рассмотрим некоторые из них:

– анимация спрайтов: среди наиболее часто используемых типов анимации в *Unity*, анимация спрайтов использует последовательное воспроизведение спрайтов для создания иллюзии движения. Вы можете создавать анимацию спрайтов, используя атласы спрайтов или анимационные клипы, которые могут меняться в зависимости от действий игрока или других событий;

– скелетная анимация предлагают расширенные возможности анимации для персонажей и объектов. Эти методы основаны на скелетных системах, состоящих из костей и связанных спрайтов или моделей. Скелетная анимация позволяет анимировать персонажей, деформировать форму и реалистичное движение.

Для создания и управления анимациями в *Unity* есть инструменты *Animator* и *Animation* они позволяют смешивать анимации, регулировать скорость и направление анимации и применять другие эффекты, такие как инверсная кинематика, чтобы сделать анимацию более естественной и реалистичной. Окна инструментов представлены на рисунках 21, 22.

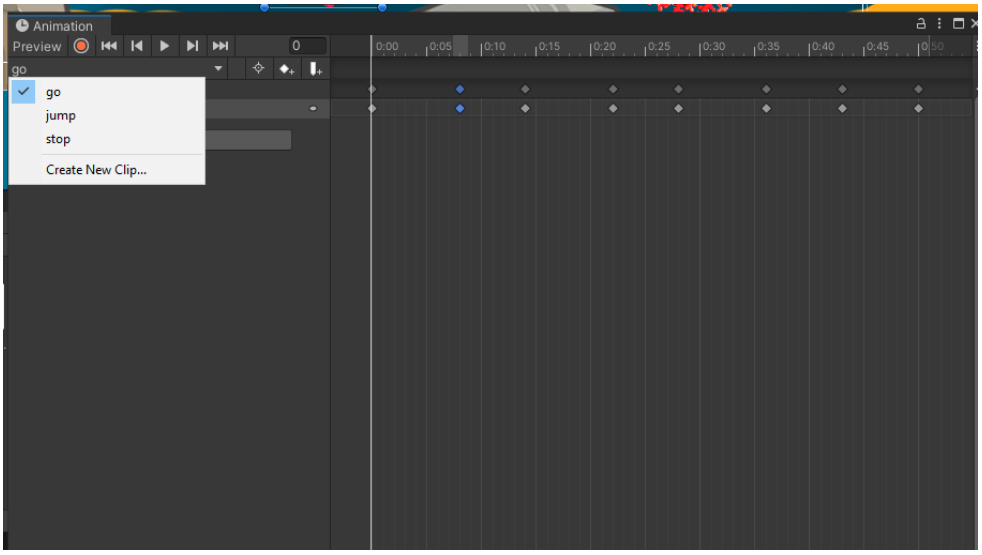


Рисунок 21 – Окно *animation*

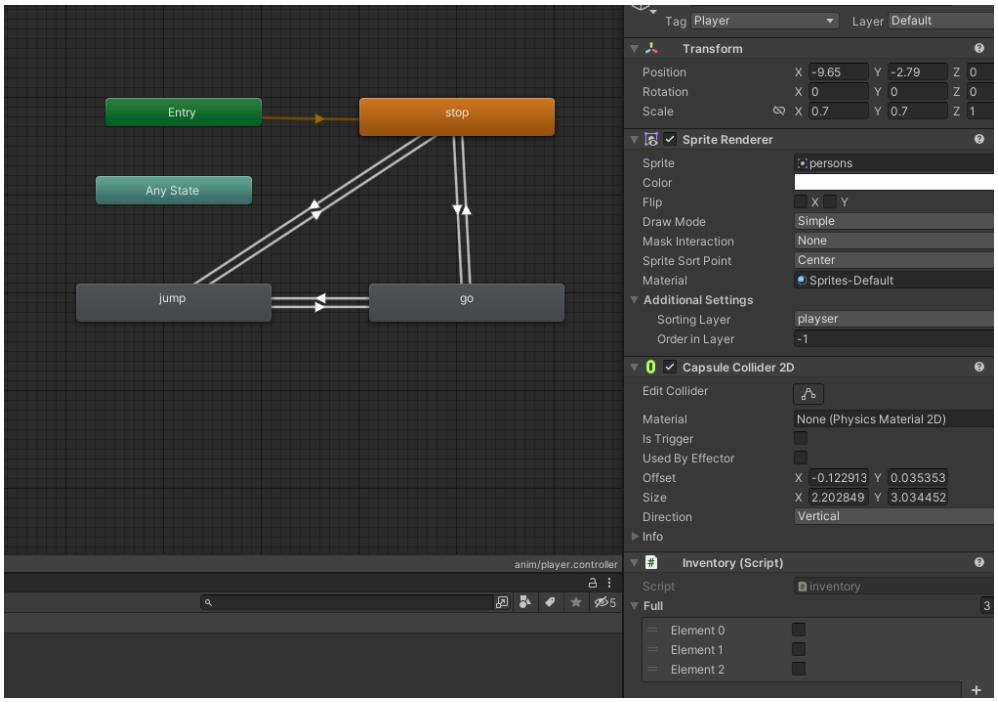


Рисунок 22 – Окно *animator*

У персонажа в игре есть 3 состояния (*stop*) – анимация покоя, (*jump*) – анимация прыжка и (*go*) – анимация движения. В параметрах окна Animator создано 2 переменные тип (*bool*). При изменении этих переменных скриптом и будут изменяться анимации персонажа.

В *Unity* существует ряд методов:

– *Start*(). Вызывается единожды при запуске сцены и используется для инициализации компонента.

– *Update*(). Вызывается каждый кадр и используется для обновления состояния компонента.

– *FixedUpdate*(). Вызывается с постоянной фиксированной частотой, независимо от кадровой частоты, и используется для физических вычислений и обновления компонента.

Для взаимодействия с объектов в *Unity* существует компонент *Collider2D* используется персонажем на всех встречающихся объектах. Этот компонент в *Unity* определяет границы и обнаруживает столкновения объектов в 2*D*-среде, определяя форму и размеры объекта, используемые для взаимодействия с другими коллайдерами в игровом мире. В результате

другие объекты могут воспринимать или пересекать область, созданную компонентом *Collider2D*. Он обеспечивает обнаружение столкновений между объектами, представляя различные формы, включая прямоугольники, круги, многоугольники или составные формы. Кроме того, компонент *Collider2D* может работать вместе с компонентом *Rigidbody2D* для создания реалистичного поведения при столкновении. *Unity* отправляет сообщения и вызывает методы обработки столкновений, когда коллайдеры объектов пересекаются или сталкиваются. Их можно использовать для реагирования на столкновения и взаимодействия с другими объектами.

Добавим главному герои эти компоненты и настроим необходимые компоненты, которые представлены на рисунке 23.

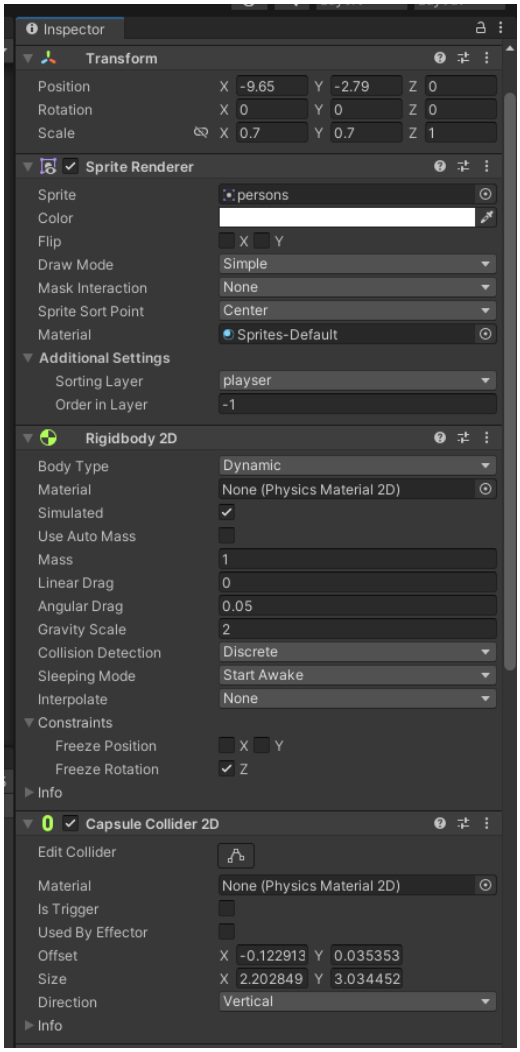


Рисунок 23 – Компоненты персонажа

Напишем необходимый скрипт для движения и изменения *bool* переменных, которые мы добавили в окне *animator*. Параметр *GetAxis* (*Horizontal*) добавим, чтобы присвоить персонажу горизонтальную скорость для движения. Функция *flip*() проверяет в какую сторону смотрит персонаж и переворачивает его если это необходимо.

При помощи метода *GetComponent* получаем доступ к полям *Animator* и *Rigidbody2D* и теперь можем менять их значения.

Для проверки приземления персонажа добавим компонент пустой *positions*. С помощью его будем проверять приземлился ли наш персонаж, пока персонаж не упал и прыгать он не сможет. Если персонаж находится на земле, то ему разрешается выполнить прыжок, устанавливая скорость *rb.velocity* по вертикали равной *Fjump* (силе прыжка) вверх.

Вернемся к созданию параллакс эффекта. Для этого необходимо создать несколько скриптов. Скрипт *ParallaxLE* (для передвижения локаций). Добавим переменную *parallaxFactor*. Она определяет скорость, с которой объект движется относительно движения камеры. Метод принимает дельта-значение, которое представляет собой изменение положения камеры.

Затем метод создает новую переменную позиции, *newPos*, которая равна текущей локальной позиции объекта. Затем вычитаем дельту, умноженную на коэффициент параллакса, из компонента *x* вектора *newPos*. По сути, это перемещает объект влево или вправо в зависимости от направления движения камеры. Метод устанавливает локальное положение объекта в новое положение, которое эффективно перемещает объект вместе с камерой, но с другой скоростью из-за фактора параллакса.

Скрипт *ParallaxCAX*, который определяет делегат и использует его для запуска метода, когда камера перемещается по оси x. В методе *Start*() скрипт инициализирует *oldPosition* текущей x-позицией камеры. В методе *Update*() скрипт проверяет, сместилась ли камера по оси x с момента последнего кадра. Если это так, сценарий вычисляет пройденное расстояние («дельта») и

запускает событие *onCameraTranslate*, передавая «дельту» в качестве параметра. Наконец, *oldPosition* обновляется до текущей x-позиции камеры.

В скрипте *ParalaxBACK* Функция *SetLayers*() перебирает дочерние объекты и добавляет в список все компоненты *ParallaxLE*. Он также переименовывает объекты в «Слой-», за которым следует индекс. Функция *Move*() вызывается при перемещении камеры. Он выполняет итерацию по списку компонентов *ParallaxLE* и вызывает их функцию *Move*(), передавая дельта-значение (т. е. насколько сдвинулась камера). Эта функция отвечает за обновление положения объектов.

Создадим скрипт *inventory* для инвентаря куда будут помещаться собранные на локации ключи. В нём будет задаваться количество слотов в нашем инвентаре.

В скрипте *pickUP* есть общедоступная переменная под названием *inventory*, которая назначается компоненту инвентаря игрока при запуске скрипта. Это предполагает, что объект игрока имеет тег Players и прикрепленный к нему компонент инвентаря. Сценарий также имеет общедоступную переменную под названием *slotButton*, которая является префабом для кнопки слота инвентаря, которая будет создана, когда предмет будет добавлен в инвентарь.

Когда игрок сталкивается с триггерной областью, скрипт проверяет каждый слот в инвентаре игрока, чтобы убедиться, что он пуст. Если он находит пустой слот, он устанавливает для свойства *full* этого слота значение *true* (указывая, что он сейчас занят) и создает новую кнопку слота инвентаря, используя префаб *slotButton*. Затем он уничтожает элемент, вызвавший столкновение, и выходит из цикла.

Если в инвентаре нет пустых слотов, скрипт ничего не делает и предмет не добавляется в инвентарь.

Создадим эффект пузырей во время прыжка. В *Unity2D* эффекты используются для добавления визуальных улучшений к игровым объектам и

сценам. Окно элемента *Effects* представлено на рисунке 24. Результат разработанного эффекта на рисунке 25.

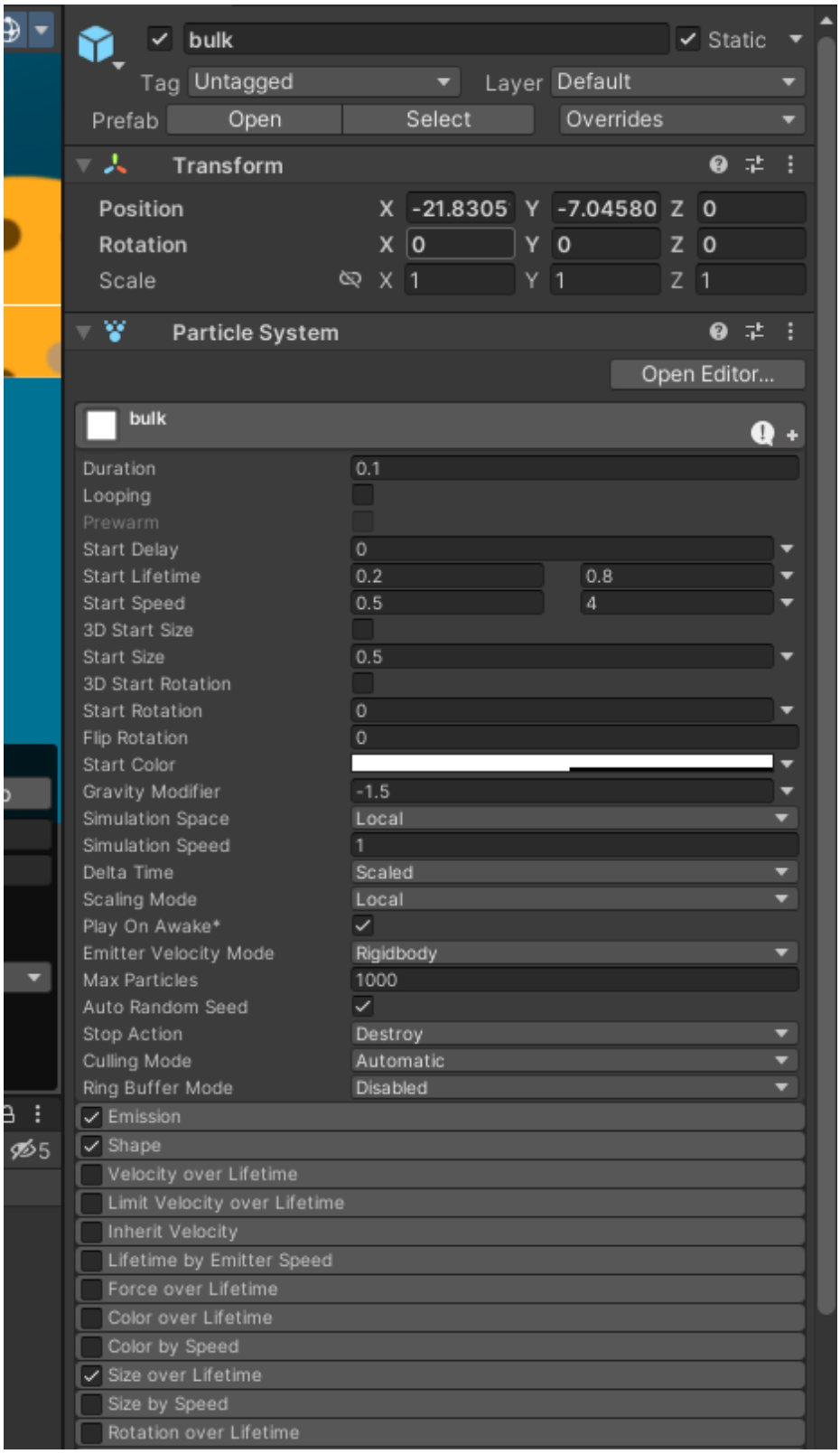


Рисунок 24 – Окно эффектов

Рисунок 25 – Эффект добавленный в игру

В ходе реализации игрового приложения были разработаны диаграмма использования игры и диаграмма деятельности. Диаграмма использования игры представлена на рисунке 26. Диаграмма деятельности представлена на рисунке 27.

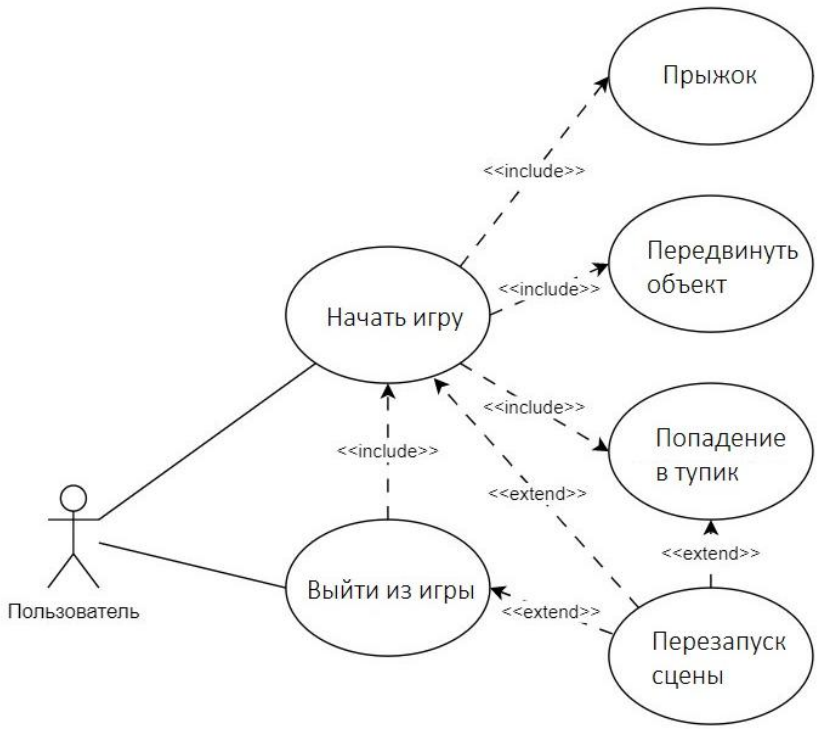


Рисунок 26 – Диаграмма использования

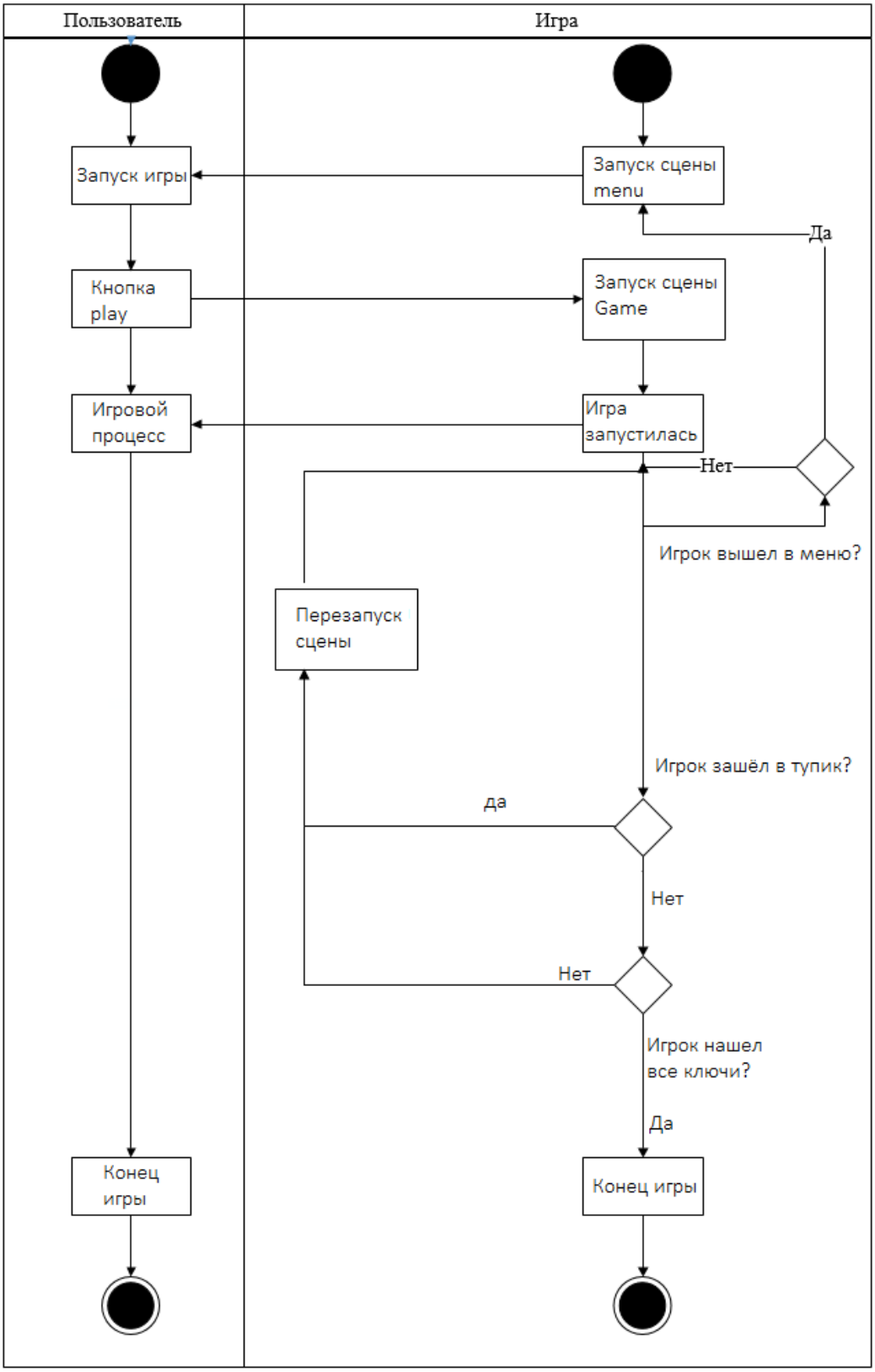


Рисунок 27 – Диаграмма деятельности

1.4 Тестирование

В процессе разработки игры тестирование имеет решающее значение для обнаружения и устранения ошибок, обеспечения функциональности и стабильности игры, а также подтверждения того, что игровой процесс соответствует требованиям. Однако на этом этапе проекта проводить обширное тестирование нецелесообразно, так как в последующей разработке будет доработана лишь небольшая часть функциональности. Результат тестирования главного меню на рисунке 28.

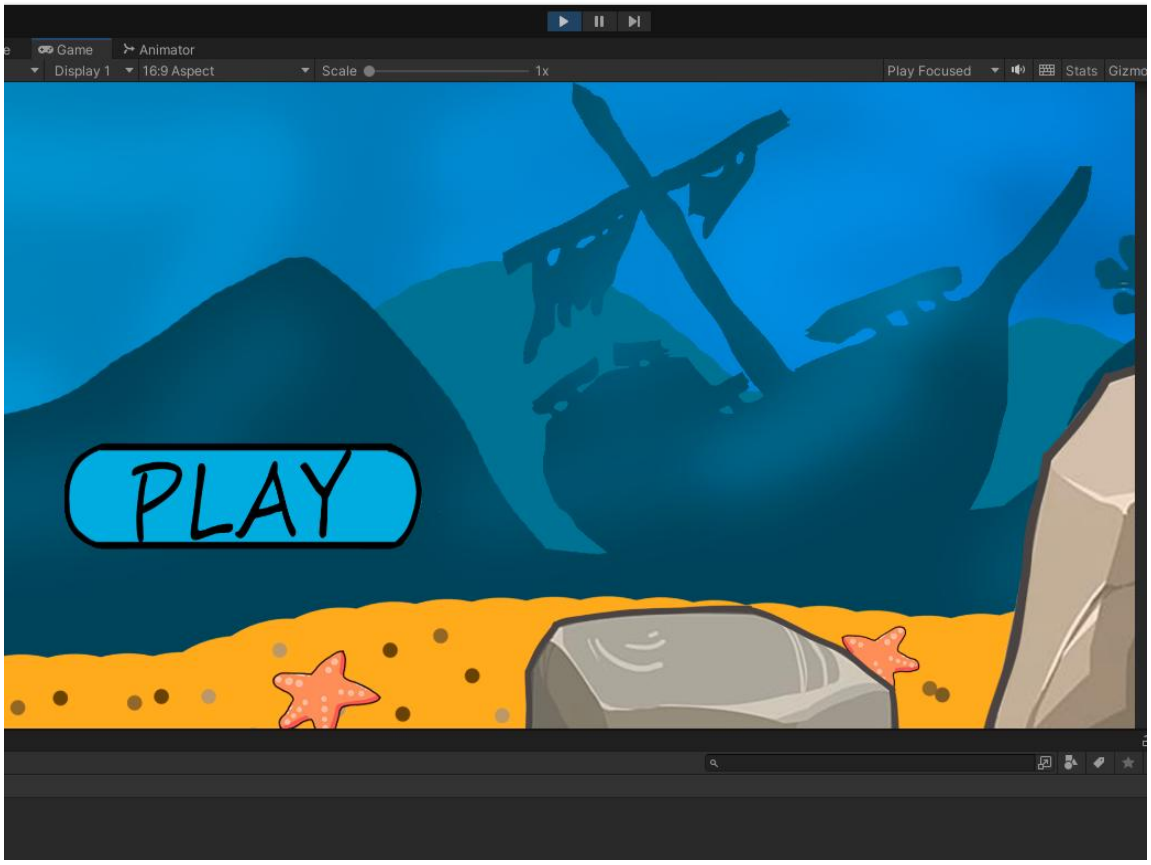


Рисунок 28 – Главное меню

Как мы видим кнопка главного меню функционирует правильно и перекидывает нас на сцену 1, которая представлена на рисунке 29.

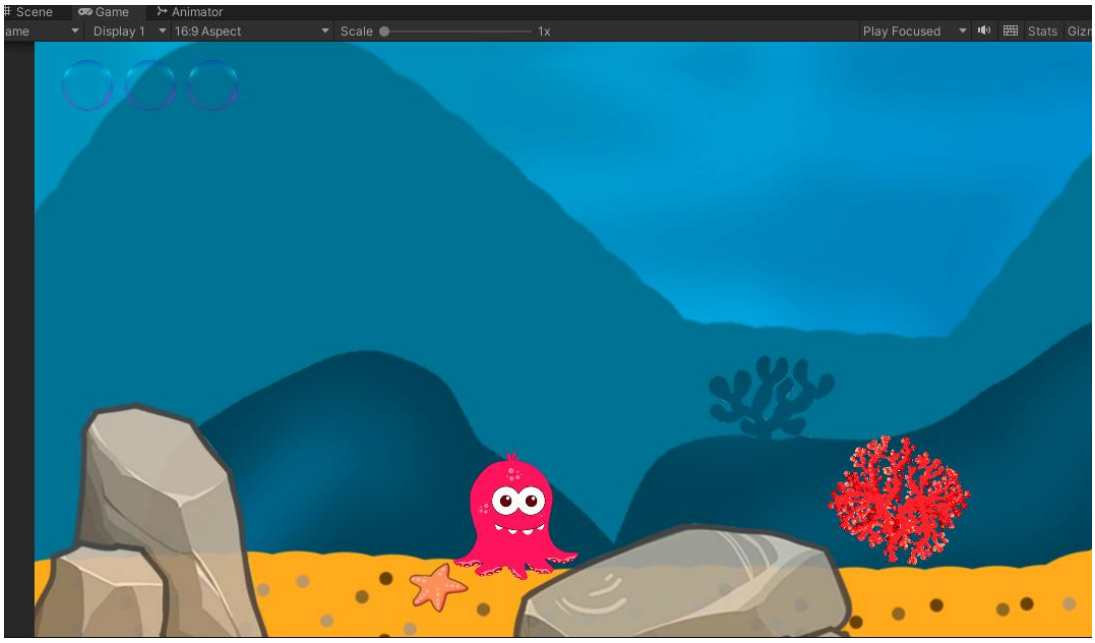


Рисунок 29 – начальная сцена

Проверили взаимодействие персонажа с объектами, всё работает правильно, результаты представлены на рисунке 30.

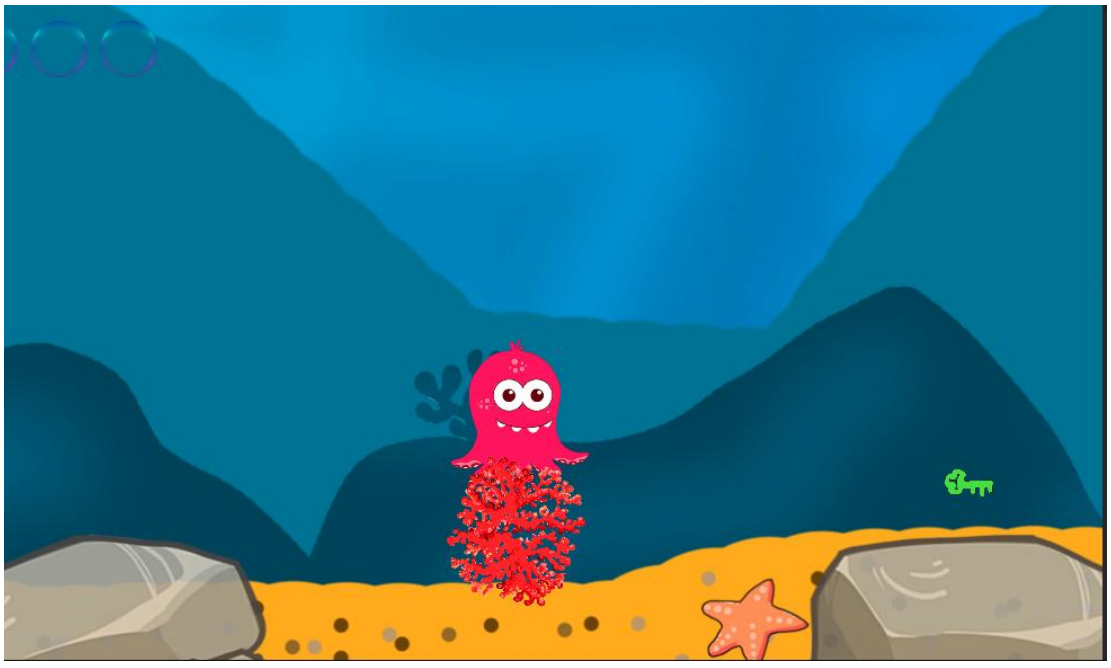


Рисунок 30 – Взаимодействие с объектами

Проверив на работоспособность функционирование инвентарь, убеждаемся, что всё работает корректно. Ключи удаляются со сцены и попадают в инвентарь. Это представлено на рисунке 31.

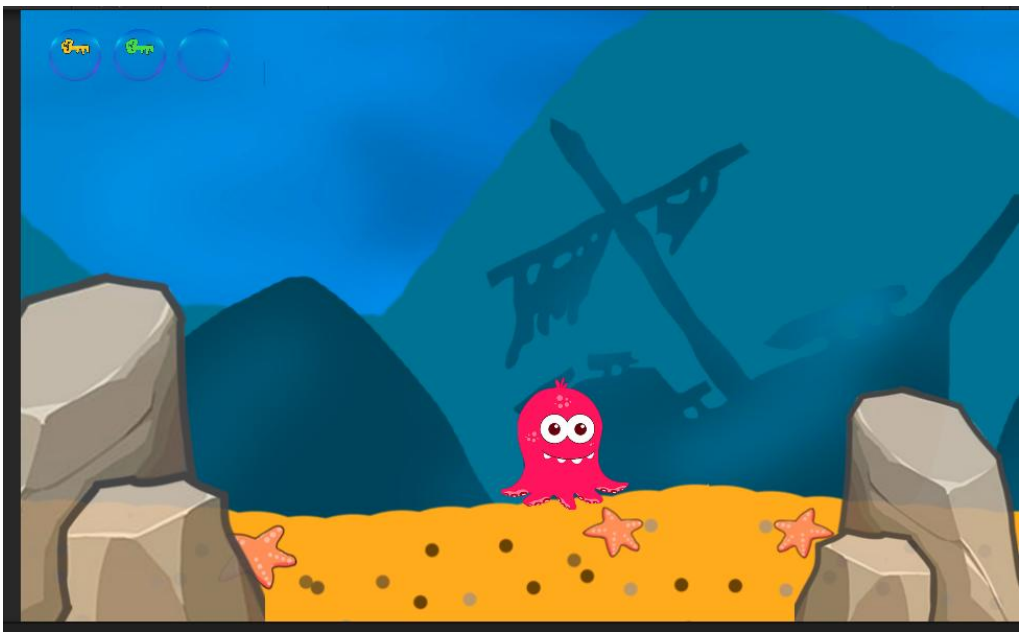


Рисунок 31 – Проверка инвентаря

Финальный вид сцены представлен на рисунке 32.

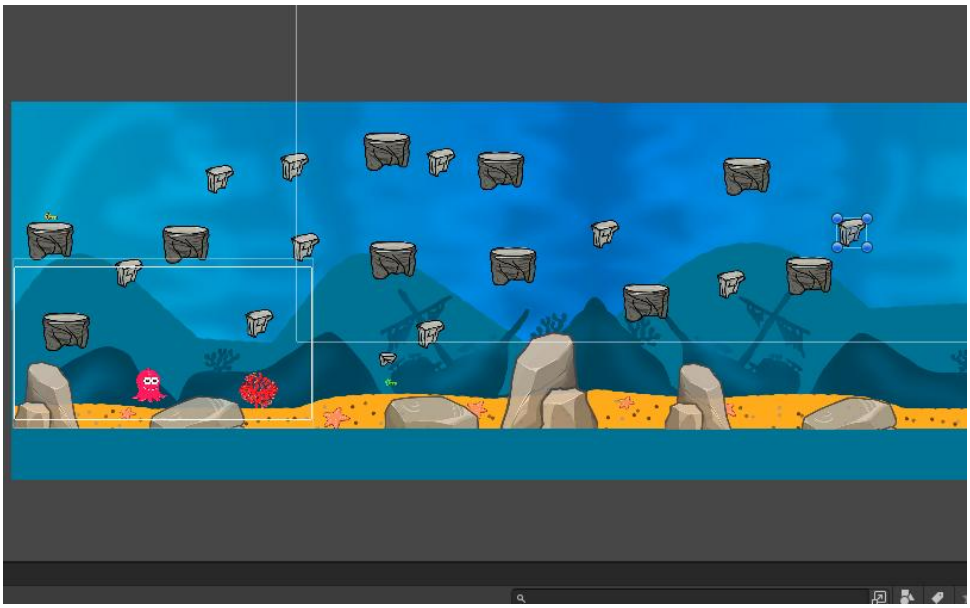


Рисунок 32 – Финальный вид сцены

На данный момент в игре реализованы основные механики при помощи которых в будущем можно добавлять все более сложные головоломки, в скором времени возможны значительные изменения игры. В дальнейшем возможна адаптация игры под смартфон.

2 Безопасность жизнедеятельности

2.1 Электробезопасность

Электрический ток оказывает различное воздействие на живую ткань. При прохождении через организм человека он оказывает тепловое, электролитическое, механическое и биологическое воздействие. Тепловое воздействие вызывает ожоги, нагревание органов, функциональные расстройства. Электролитический эффект вызывает разложение органической жидкости и нарушение ее состава. Расслоение и разрыв тканей тела происходит из-за механического воздействия. К биологическим эффектам относятся раздражение, возбуждение и нарушение внутренних биологических процессов.

Электротравмы могут быть общими (поражение электрическим током с судорогами, остановкой дыхания и сердечной деятельности) и местными (ожоги, металлизация кожи, электроофтальмия). Такие факторы, как сила тока, время прохождения, характеристики тока и путь тока, влияют на исход поражения электрическим током.

Сопротивление играет существенную роль в определении величины тока, проходящего через тело. Внутреннее сопротивление человека не превышает нескольких сотен Ом. На сопротивление организма воздействию электрического тока оказывает влияние физическое и психическое состояние человека. Не-здоровье, утомление, голод, опьянение, эмоциональное возбуждение приводят к снижению сопротивления.

Максимальный ток, с которым человек может справиться самостоятельно и вырваться из электрической цепи, зависит от продолжительности и скорости прохождения тока через тело. Если ток длится 10 секунд или менее, допустимый ток составляет 6 мА, а более 10 секунд - 2 мА. Когда пострадавший не может оторваться от токоведущих частей, ток считается не отпускающим. Постоянный ток более опасен, чем

переменный ток при высоком напряжении выше 500 В. Среди различных путей прохождения тока через тело наиболее опасным является через голову и руки, голову и ноги, а также руки и ноги, которые воздействуют на головной мозг. , сердце и легкие. Более высокие температуры и влажность повышают риск поражения электрическим током, поскольку влага снижает сопротивление кожи. Предельно допустимые контактные напряжения и токи, протекающие через тело человека при аварийной работе электроустановок постоянного и переменного тока частотой 50 и 400 Гц, установлены ГОСТ 12.1.038-82* бытового назначения постоянного и переменного тока частотой 50 и 400 Гц.

2.2 Требования к освещению рабочего места

Избегайте бликов. Блики на экране компьютера могут вызывать напряжение глаз, поэтому важно расположить компьютер так, чтобы он не был обращен к окну или другому источнику света, который может отражаться на экране.

Используйте рабочее освещение. Рабочее освещение, направленное на рабочую поверхность, может помочь снизить нагрузку на глаза и улучшить видимость. Этого можно добиться с помощью настольной лампы или потолочных светильников с регулируемыми лампочками.

Учитывайте цветовую температуру: цветовая температура (измеряемая в Кельвинах) освещения может влиять на то, насколько комфортно вы себя чувствуете во время работы. Холодный белый свет (от 5 000 до 6 500 К) может помочь вам сохранять бдительность и концентрацию, а теплый свет (от 2 700 до 3 000 К) может создать расслабляющую атмосферу.

Используйте диммеры. Диммеры помогут вам отрегулировать яркость освещения в соответствии с вашими потребностями и снизить нагрузку на глаза. Избегайте мерцания. Мерцание света или экрана может вызвать

головную боль и напряжение глаз. Ищите светодиодные фонари или мониторы с более высокой частотой обновления, чтобы избежать мерцания.

2.3 Правильное положение за столом

По мнению ортопедов, поддержание хорошей осанки во время сидения имеет решающее значение, и неправильное положение ног может этому помешать. Даже при постоянных усилиях держать спину прямо поддерживать позвоночник практически невозможно без правильного положения ног. Скрещивание ног во время длительного сидения может вызвать искривление позвоночника, а неправильное положение ног может повлиять на кровообращение, что приведет к кислородному голоданию и усталости. Чтобы предотвратить это, для правильного положения ног рекомендуются ортопедические подставки для ног, хотя их следует приобретать только у надежных производителей. Важно привыкнуть к удобному положению ног либо на земле, либо на подножке, и соответствующим образом отрегулировать высоту стула. Качественные стулья и правильная осанка могут предотвратить вредные проблемы опорно-двигательного аппарата, такие как искривление позвоночника, радикулит и остеохондроз.

Когда вы сидите за компьютером в течение длительного времени, важно соблюдать правильную позу, чтобы избежать напряжения мышц и проблем со здоровьем. Вот несколько рекомендаций по правильной позе за компьютерным столом. Регулируйте высоту стула таким образом, чтобы ваша бедренная кость была параллельна полу, а колени находились на уровне бедер или ниже. Обеспечьте поддержку для спины с помощью специального офисного стула или подушки для поясницы. Ваша спина должна быть прямой и легко поддерживаться. Регулируйте высоту монитора таким образом, чтобы верхняя часть экрана была на уровне глаз. Это поможет предотвратить напряжение в шее и глазах.

Рукавицы должны лежать на поверхности стола, а локти - в углах 90 градусов. Ваши запястья и руки должны быть прямыми и расслабленными. Если вы работаете на клавиатуре, то расположите ее перед собой на уровне локтей. Руки должны быть расположены прямо и расслаблены.

Периодически делайте перерывы в работе, чтобы размять мышцы и глаза. Встаньте и прогуляйтесь, посмотрите за окно, закройте глаза на несколько минут.

Заключение

В результате выполнения выпускной квалификационной работы была разработана игры в жанре Головоломка. Проведен анализ других игр головоломок, сравнение их особенностей и постановка задач на разработку собственной игры. В ходе разработки был получен опыт разработки игр на платформе *Unity*.

В результате разработки игрового приложения можно сделать следующие вывод, что комбинация *Unity*, *Photoshop* и *C#* позволила быстро разработать игру и также не потерять высокую производительность и надежность.

В будущем игровое приложение может изменяться согласно запросам пользователей.

Список использованных источников

1. Разработка игр на Unity [Электронный ресурс]. *URL*: *https://tproger.ru/articles/top-20-sovetov-i-trjukov-kotorye-pomogut-nachinajushhim-v-unity-3d/* (дата обращения: 29.04.2023).

2. Как создать игру на Unity [Электронный ресурс]. *URL*: *https://www.sravni.ru/kursy/info/kak-sozdat-igru-na-unity/* (дата обращения: 10.05.2023).

3. Движок *Unity* – особенности преимущества и недостатки [Электронный ресурс]. *URL*: *https://cubiq.ru/dvizhok-unity/* (дата обращения: 13.05.2023).

4. *Unity Forums* [Электронный ресурс]. *URL*: https://forum.unity.com/. (дата обращения: 19.05.2023).

5. Безопасность жизнедеятельности. /Под ред. Н.А. Белова - М.: Знание, 2000 - 364с. (дата обращения: 14.06.2023).

Приложение А

(обязательное)

Развивающее приложение на *Unity* в жанре «головоломка»

Листинг программы

Файл *playerDVIJ.cs*

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class playerDVIJ : MonoBehaviour
{


    public float speed;
    public float Fjump;
    private float move;
    private bool povorot = true;
    private Rigidbody2D rb;

    private bool pol;
    public Transform positions;
    public float checkRadiys;
    public LayerMask whatPosition;
    private Animator anim;
    public GameObject effects;
    private void Start()
    {
        rb = GetComponent<Rigidbody2D>();
        anim = GetComponent<Animator>();
    }

    private void FixedUpdate()
    {
        move = Input.GetAxis("Horizontal");
        rb.velocity = new Vector2(move * speed, rb.velocity.y);
        if (povorot==false&& move > 0)
        {
            Flip();
        }else if(povorot==true && move<0){
            Flip();

        }

        if (move == 0)
        {
            anim.SetBool("runbool", false);
        }
        else
        {
            anim.SetBool("runbool", true);
        }
    }

    private void Update()
```

```
    {
        pol = Physics2D.OverlapCircle(positions.position,
checkRadiys, whatPosition);
        if (pol == true && Input.GetKeyDown(KeyCode.Space)) {
            rb.velocity = Vector2.up * Fjump;
            Instantiate(effects, transform.position,
Quaternion.identity);
        };
        if (pol == true)
        {
            anim.SetBool("jumpbool", false);

        }
        else
        {

                anim.SetBool("jumpbool", true);

        }
    }

    void Flip()
    {
        povorot = !povorot;
        Vector3 Scaler = transform.localScale;
        Scaler.x *= -1;
        transform.localScale= Scaler;
    }

}
```

Файл *pickUP.cs*

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class pickUP : MonoBehaviour
{

    public inventory inventory;
    public GameObject slotButton;

    private void Start()
    {

inventory=GameObject.FindGameObjectWithTag("Player").GetComponen
t<inventory>();

    }
```

```
    private void OnTriggerEnter2D(Collider2D other)
    {
        if (other.CompareTag("Player"))
        {
            for(int i=0; i < inventory.slots.Length; i++)
            {
                if (inventory.full[i]== false)
                {
                    inventory.full[i] = true;
                    Instantiate(slotButton,
inventory.slots[i].transform);
                    Destroy(gameObject);
                    break;
                }
            }
        }
    }
}
```

Файл *slot.cs*

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class slot : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }
}
```

Файл *ParallaxLE.cs*

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
[ExecuteInEditMode]
public class ParallaxLE : MonoBehaviour
{


        public float parallaxFactor;
```

```
        public void Move(float delta)
        {
            Vector3 newPos = transform.localPosition;
            newPos.x -= delta * parallaxFactor;
            transform.localPosition = newPos;
        }
    }
```

Файл *ParallaxCAM.cs*

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
[ExecuteInEditMode]
public class ParallaxCAM : MonoBehaviour
{



        public delegate void ParallaxCAMDelegate(float
deltaMovement);
        public ParallaxCAMDelegate onCameraTranslate;
        private float oldPosition;
        void Start()
        {
            oldPosition = transform.position.x;
        }
        void Update()
        {
            if (transform.position.x != oldPosition)
            {
                if (onCameraTranslate != null)
                {
                    float delta = oldPosition -
transform.position.x;
                    onCameraTranslate(delta);
                }
                oldPosition = transform.position.x;
            }
        }

}
```

Файл *ParallaxBACK.cs*

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ParalaxBACK : MonoBehaviour
{
```

```
    [ExecuteInEditMode]


        public ParallaxCAM parallaxCAM;
        List<ParallaxLE> parallaxLE = new List<ParallaxLE>();

        void Start()
        {
            if (parallaxCAM == null)
                parallaxCAM =
Camera.main.GetComponent<ParallaxCAM>();
            if (parallaxCAM != null)
                parallaxCAM.onCameraTranslate += Move;
            SetLayers();
        }

        void SetLayers()
        {
            parallaxLE.Clear();
            for (int i = 0; i < transform.childCount; i++)
            {
                ParallaxLE layer =
transform.GetChild(i).GetComponent<ParallaxLE>();

                if (layer != null)
                {
                    layer.name = "Layer-" + i;
                    parallaxLE.Add(layer);
                }
            }
        }
        void Move(float delta)
        {
            foreach (ParallaxLE layer in parallaxLE)
            {
                layer.Move(delta);
            }
        }
}
```

Файл *inventory.cs*

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class inventory : MonoBehaviour
{
    public bool[] full;
    public GameObject[] slots;
}
```

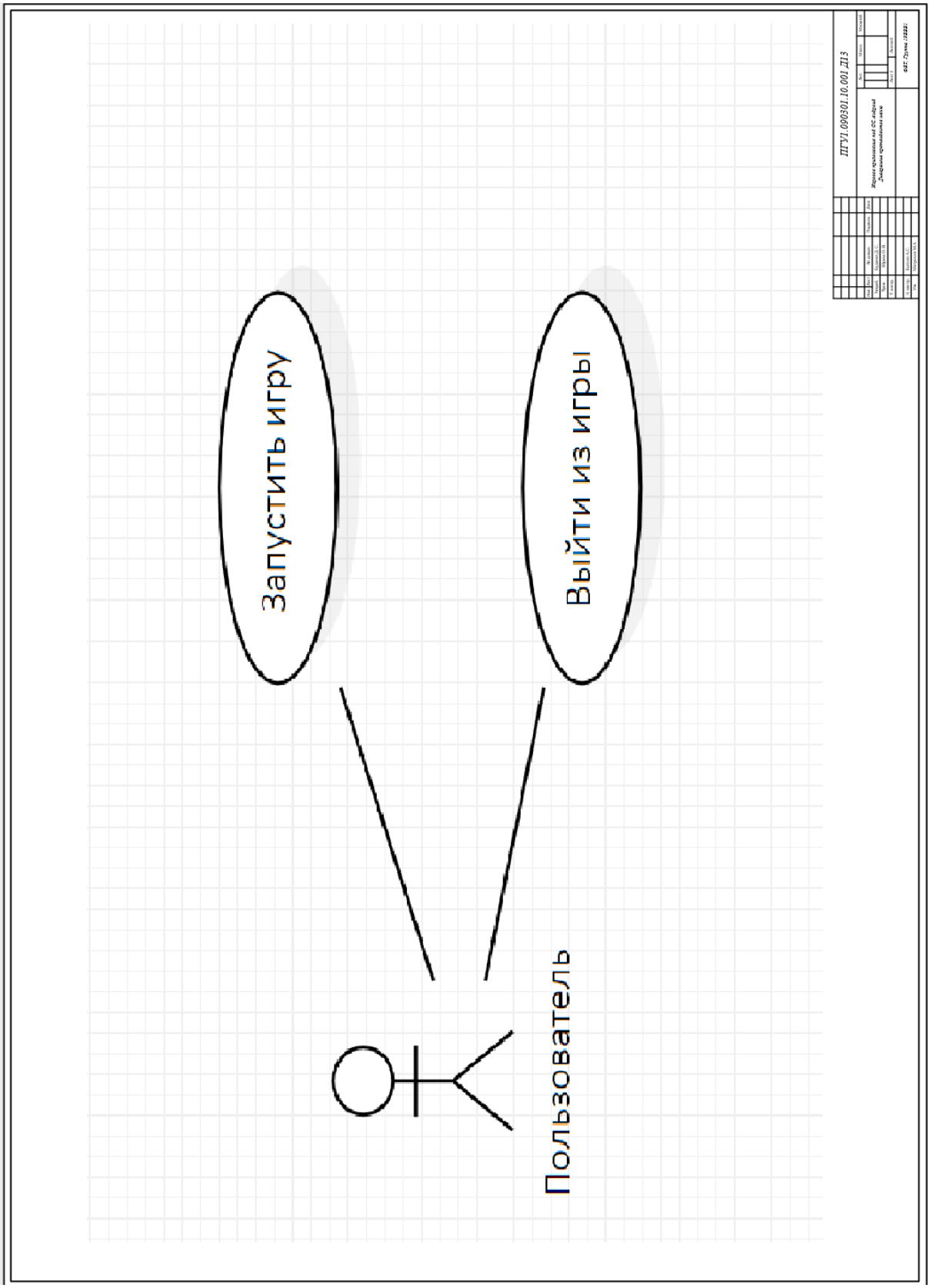
Приложение Б

(обязательное)

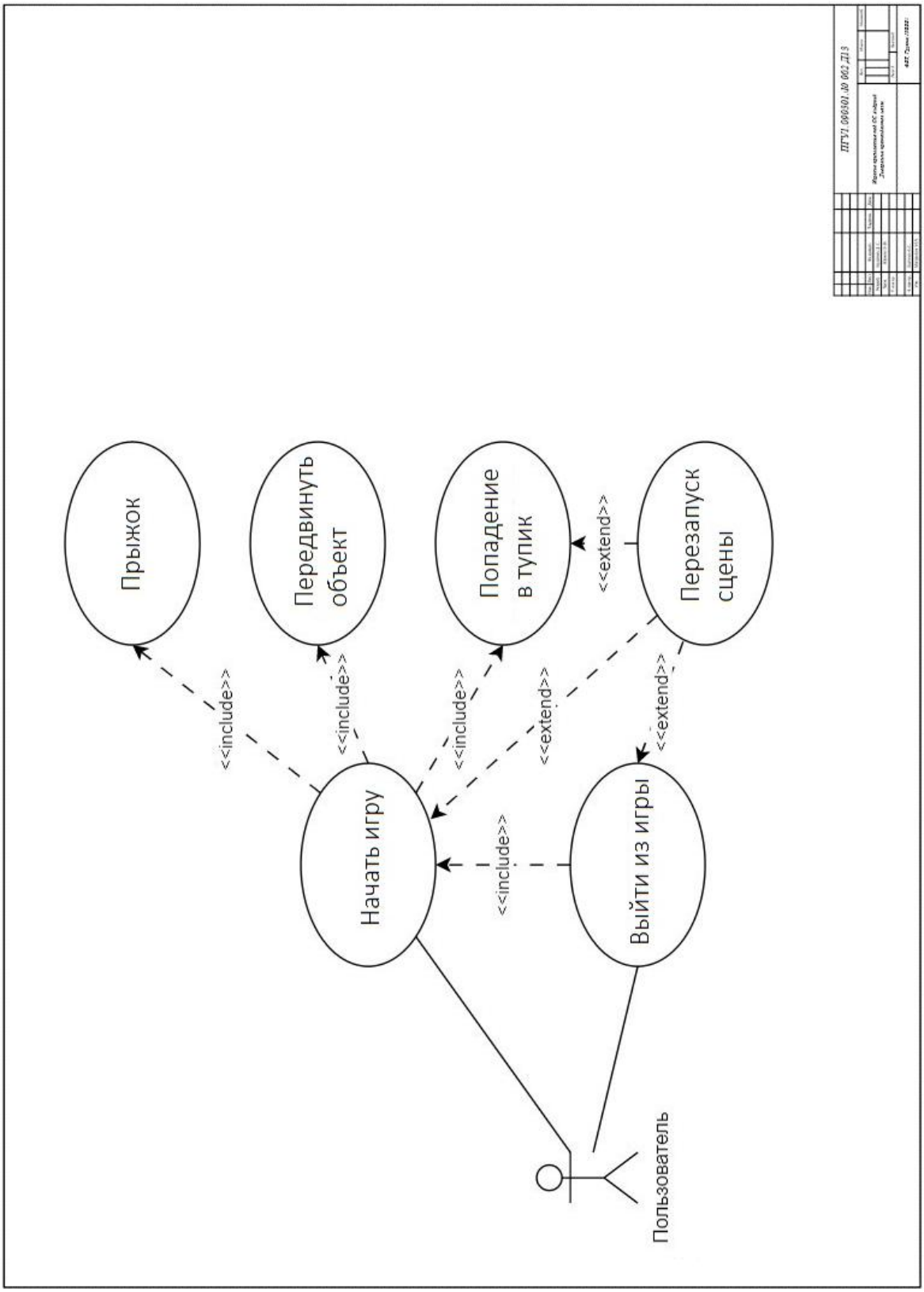
Развивающее приложение на *Unity* в жанре «головоломка»

UML – диаграммы

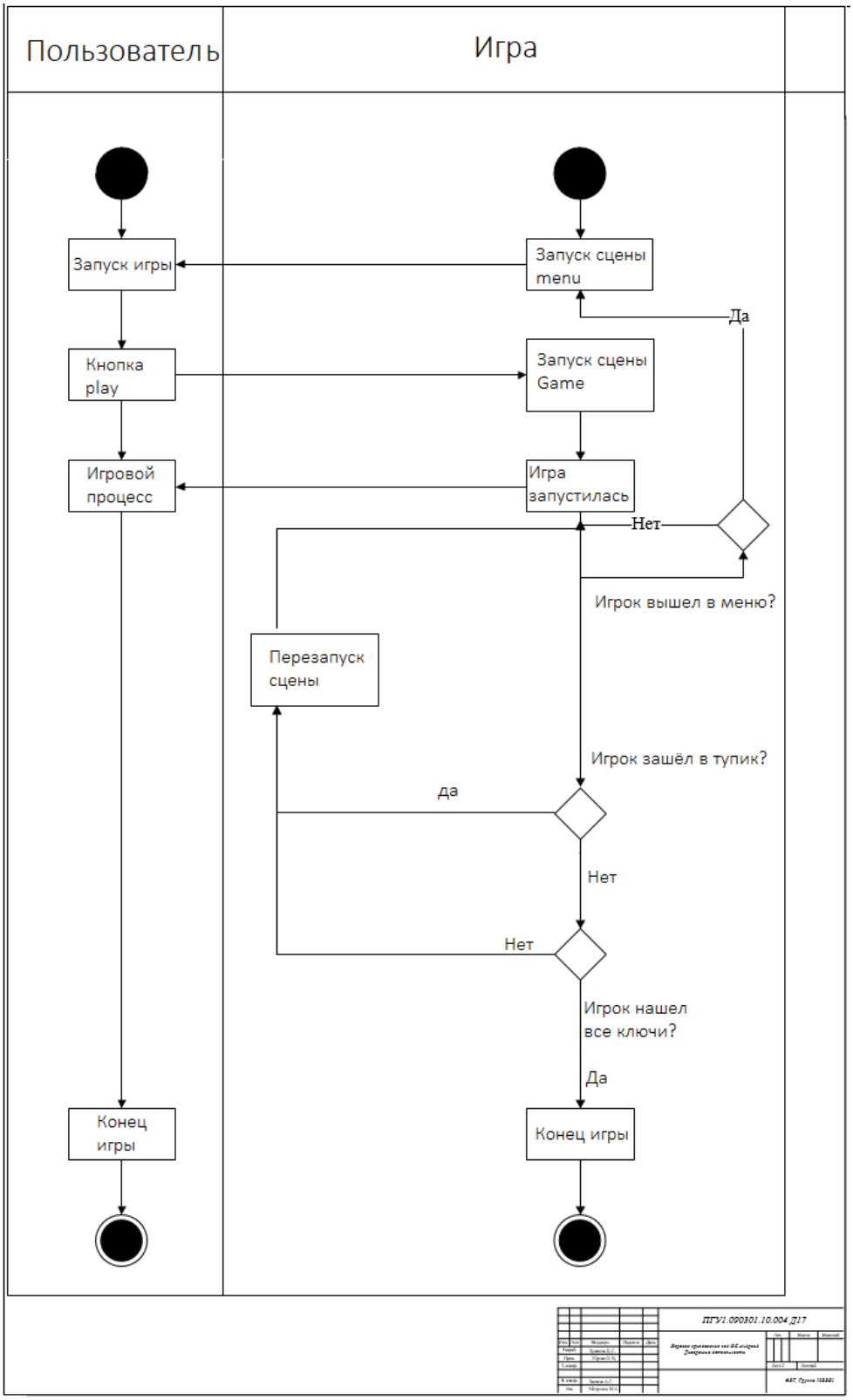
Приложение Б. 1 – *UML* – диаграмма претендентов меню



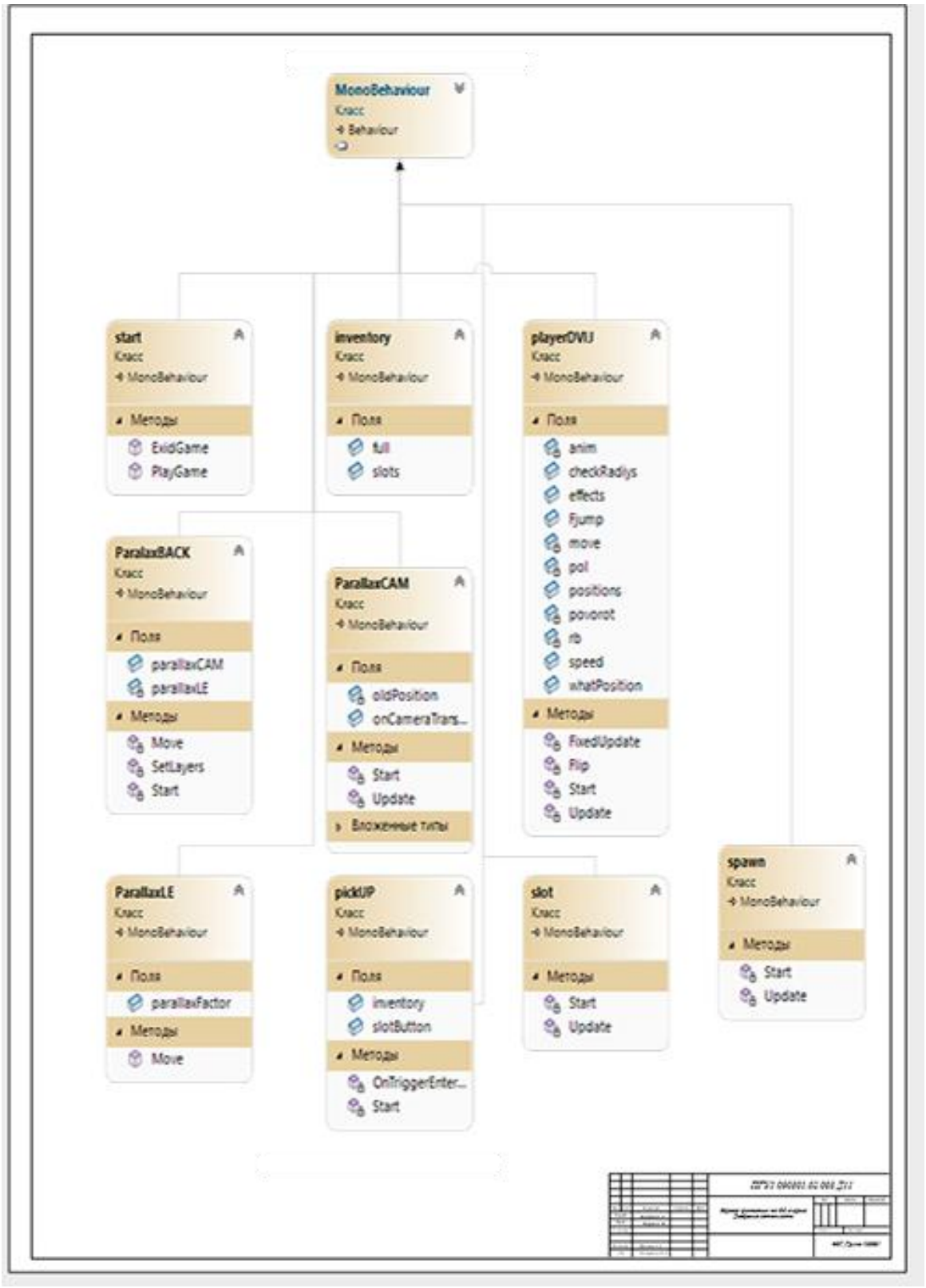
Приложение Б. 2 – *UML* – диаграмма претендентов



Приложение Б. 3 – *UML* – диаграмма деятельности



Приложение Б. 4 – *UML* – диаграмма классов



Приложение В

(обязательное)

Развивающее приложение на *Unity* в жанре «головоломка»

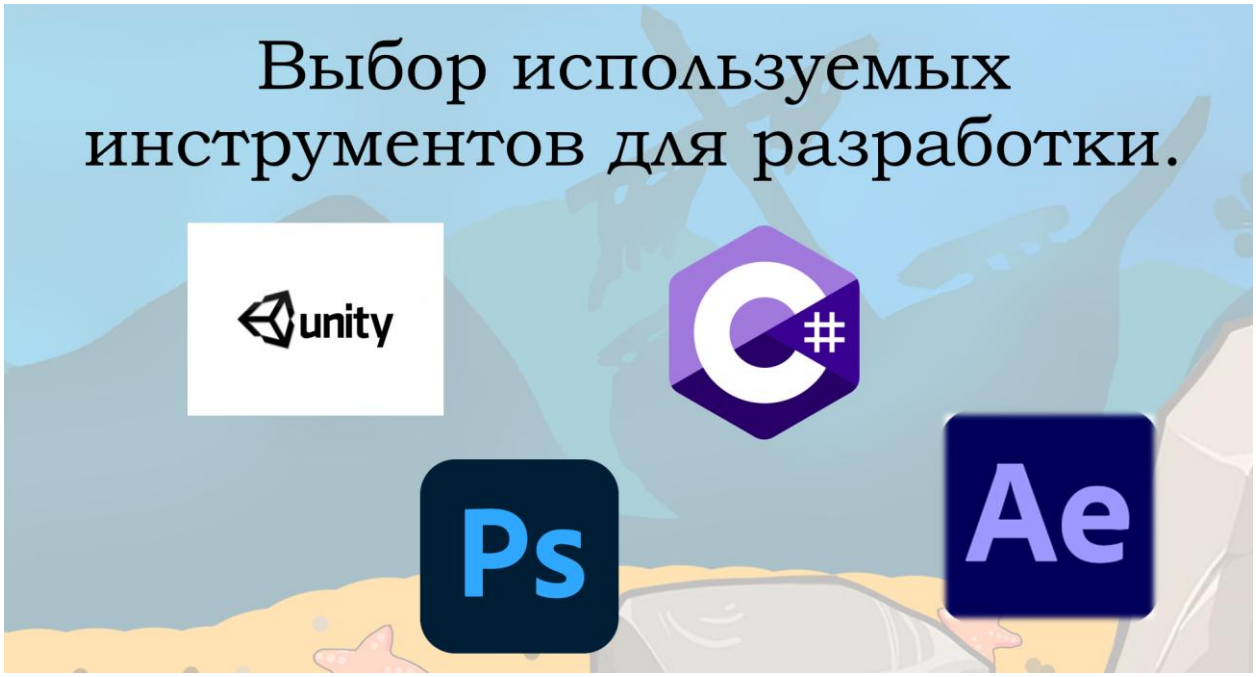
Презентация

Развивающее приложение на Unity в жанре «головоломка»

Автор:
Студент группы 19ВВВ1 Кочетков К.А.
Научный руководитель:
Акифьев И. В.

Популярность игр Головоломок

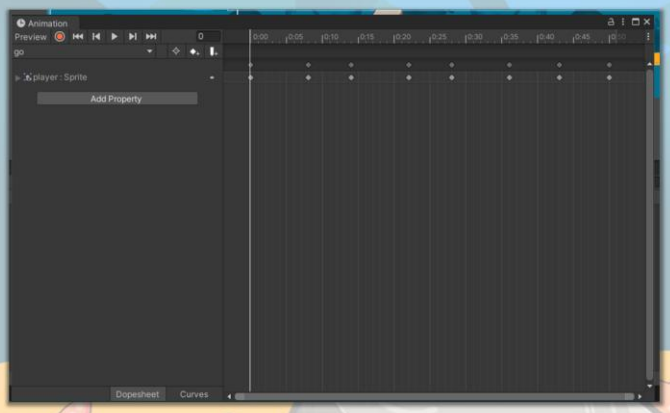
Цель:
Разработать игровое приложение с понятным интерфейсом и привлекательной графикой.
Разработать анимации для персонажа.
Написать сценарий для игры.


Выбор используемых инструментов для разработки.
unity
C#
Ps
Ae

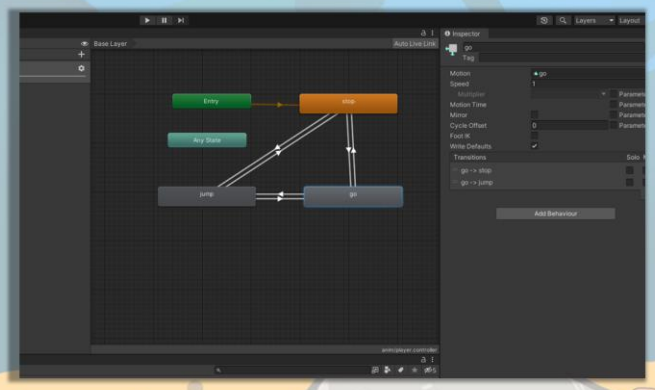
Внешний вид главного персонажа созданного при помощи редактора PhotoShop

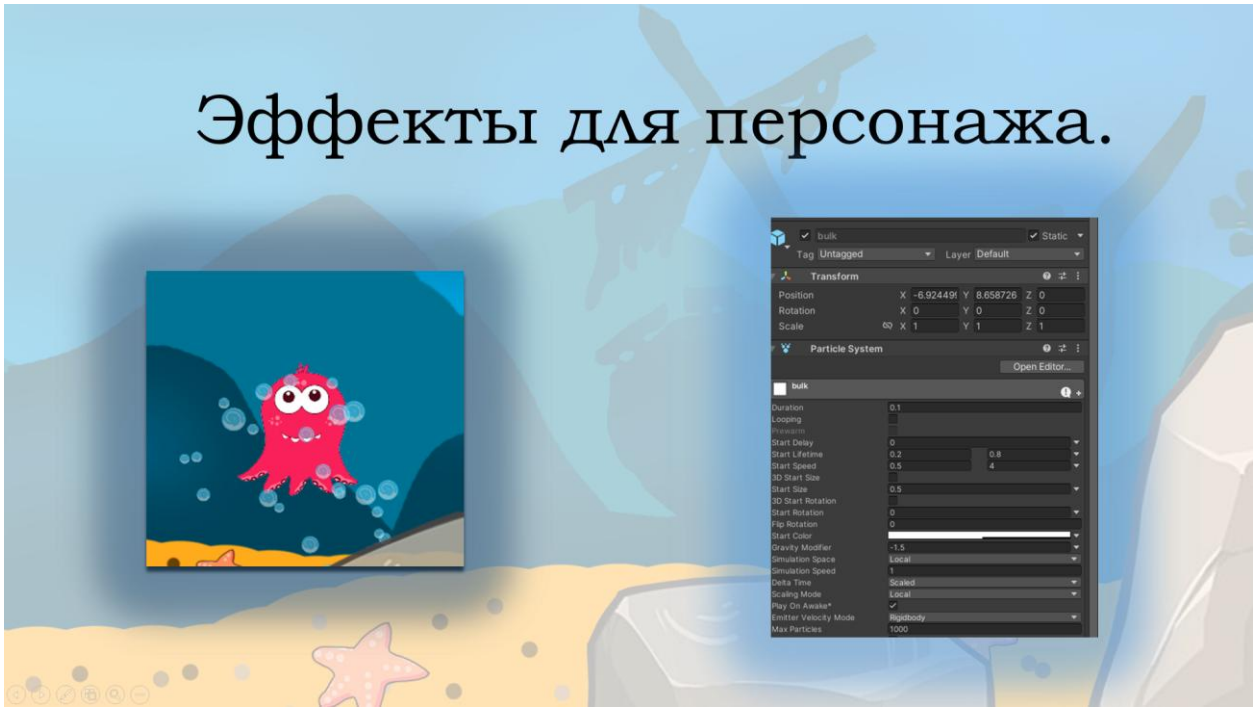
Создание спрайтовой анимации при помощи After Effects

Реализация анимации в Unity.(Animatiom)



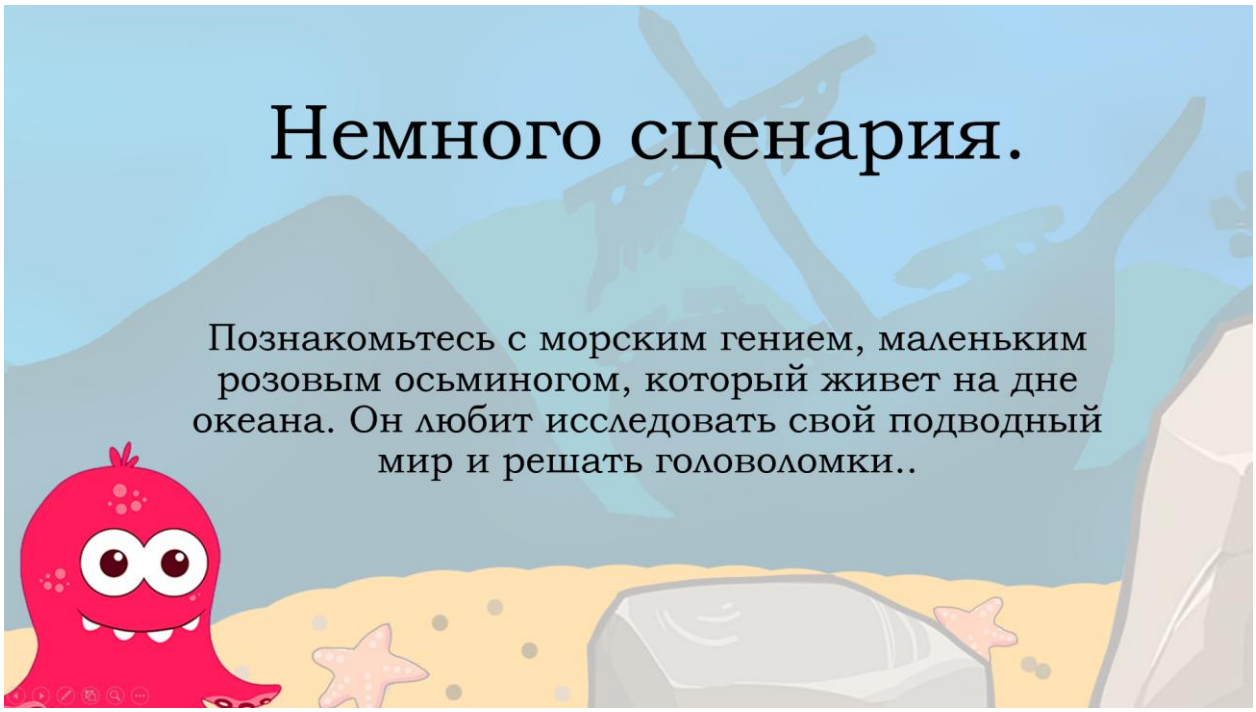
(Animatior)




Эффекты для персонажа.
bulk
Static
Tag Untagged
Layer Default
Transform
Position
Rotation
Scale
Particle System
Open Editor...
bulk
Duration
Looping
Prewarm
Start Delay
Start Lifetime
Start Speed
3D Start Size
Start Size
3D Start Rotation
Start Rotation
Flip Rotation
Start Color
Gravity Modifier
Simulation Space
Simulation Speed
Delta Time
Scaling Mode
Play On Awake*
Emitter Velocity Mode
Max Particles

Готовая сцена

Простое главное меню.
PLAY


Немного сценария.
Познакомьтесь с морским гением, маленьким розовым осьминогом, который живет на дне океана. Он любит исследовать свой подводный мир и решать головоломки..

Финальная версия игры
на данный момент:

Спасибо за внимание!